UNCERTAINTY QUANTIFICATION FOR HOLOGRAPHIC INTERFEROGRAPHIC IMAGES

by

Laurie Ann Centauri

A thesis submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Chemical Engineering

The University of Utah

December 2010

The University of Utah Graduate School

STATEMENT OF THESIS APPROVAL

The thesis of **Laurie Ann Centauri**

has been approved by the following supervisory committee members:

Eric Eddings	, Chair	**8/27/2010** Date Approved
Terry Ring	, Member	**8/27/2010** Date Approved
Jeremy Thornock	, Member	**8/27/2010** Date Approved

and by **JoAnn Lighty**, Chair of the Department of **Chemical Engineering**

and by Charles A. Wight, Dean of The Graduate School.

ABSTRACT

Current comparison methods for experimental and simulated holographic interferometric images are qualitative in nature. Previous comparisons of holographic interferometric images with computational fluid dynamics (CFD) simulations for validation have been performed qualitatively through visual comparison by a data analyst. By validating the experiments and CFD simulations in a quantifiable manner using a consistency analysis, the validation becomes a repeatable process that gives a consistency measure and a range of inputs over which the experiments and CFD simulations give consistent results.

The quantification of uncertainty in four holographic interferometric experiments was performed for use in a data collaboration with CFD simulations for the purpose of validation. The model uncertainty from image-processing, the measurement uncertainty from experimental data variation, and the scenario uncertainty from the bias and parameter uncertainty was quantified. The scenario uncertainty was determined through comparison with an analytical solution at the helium inlet (height, $x = 0$), including the uncertainty in the experimental parameters from historical weather data. The model uncertainty was calculated through a Box-Behnkin sensitivity analysis on three image-processing code parameters. Measurement uncertainty was determined through a statistical analysis to determine the time-average and standard deviation in the interference fringe positions. An experimental design matrix of CFD simulations was

performed by Weston Eldredge using a Box-Behnkin design with helium velocity, temperature, and air co-flow velocity as parameters in conjunction to provide simulated measurements for the data collaboration Data set.

Over 3,200 holographic interferometric images were processed through the course of this study. When each permutation of these images is taken into account through all the image-processing steps, the total number of images processed is over 13,000. Probability distribution functions were plotted for each interference fringe order at each measurement height, making a total of 22 PDFs. Model, scenario, and measurement uncertainty was quantified in the experiments. The CFD simulations were performed. The final uncertainty attributed to the experiments resulted in a maximum uncertainty of -7.96 fringes. The largest contributor to uncertainty was the scenario uncertainty with measurement uncertainty as the second largest. The model uncertainty was very small and as such had the smallest contribution to the overall experimental uncertainty. In the future, the results of this study will be used in conjunction with the CFD simulations discussed and their attributed error in a data collaboration to determine Data set consistency for validation.

TABLE OF CONTENTS

INTRODUCTION AND STATE OF THE ART

The development of a computational fluid dynamics (CFD) simulation of the heat flux to a ten to twenty meter transportation fuel drum in a pool fire is in progress at the University of Utah's Center for Accidental Fires and Explosions (CSAFE). This CFD simulation will help in fire protection engineering applications to determine new safety regulations for situations where hazardous, flammable materials are under high heat flux from pool fires and could explode, damaging personnel and the surroundings. For this code to be useful for its intended purposes, it must be validated. To achieve full validation of this large predictive CFD simulation, smaller, more simple CFD simulations must first be validated against simple experiments. These small CFD simulations are laid out in a structure called a hierarchy where each level relies on the validated level below to provide accurate CFD simulations. The overall CFD simulation hierarchy is shown in Figure 1. One such small-scale CFD simulation on the coupled problems level in Figure 1 is a buoyancy–driven helium plume which simply models plume fluid dynamics (i.e., entrainment, the puffing frequency, and the density gradients in the plume).

The fluid dynamic behavior of buoyant plumes is of great interest for CFD simulations due to the puff cycle exhibited by the flow. The cause of this instability is still under debate [1]. A recent study [2] theorized that puffing is due to a Rayleigh-

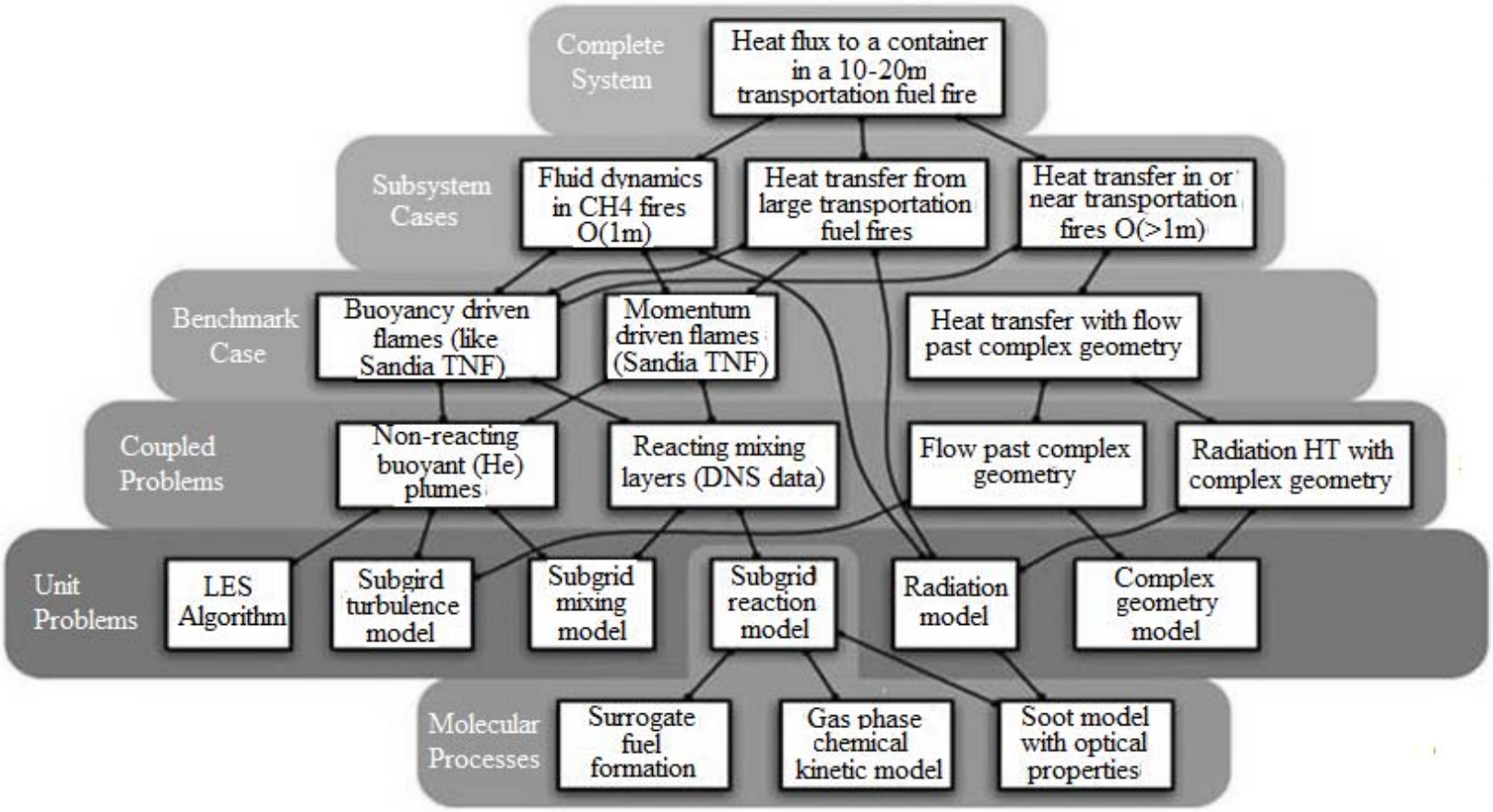


Figure 1: CFD simulation hierarchy for the University of Utah transportation fire.

Taylor instability at the interface of the air and helium. A Rayleigh-Taylor instability occurs in a system where a less dense fluid like helium advects into a more dense fluid like air [3].

Holographic Interferometry

Holographic interferometry, the analysis of a system using interfering holograms, is effective for observing transparent flows where density gradients occur in momentum transfer [4]. Holographic interferometry uses two holograms, a hologram of the air (reference hologram) and a hologram of the object (object hologram), to visualize the system nonintrusively [5]. Where the two holograms are out of phase, they destructively interfere, creating dark lines. These dark lines are called interference fringes and denote gradients of refractive index and density within the plume [6]. An original interferogram is shown in Figure 2 where the density gradients are seen through the interference fringes like a contour map; the higher the density or refractive index gradient at a height in the plume, the larger number of fringes observed. For example, at the top of the port (the bottom of the image), where the helium enters the system, a large amount of fringes are observed. The refractive index gradient is largest here (hence the largest number of interference fringes) because no mixing has occurred and the system is pure helium inside the plume whereas at the top of the interferogram, fewer fringes are seen because the entrainment of air has decreased the refractive index gradient. By measuring the locations of these interference fringes, the refractive index or density gradients can be determined and compared with those from CFD simulations for model validation purposes, after determining their time-averaged values.

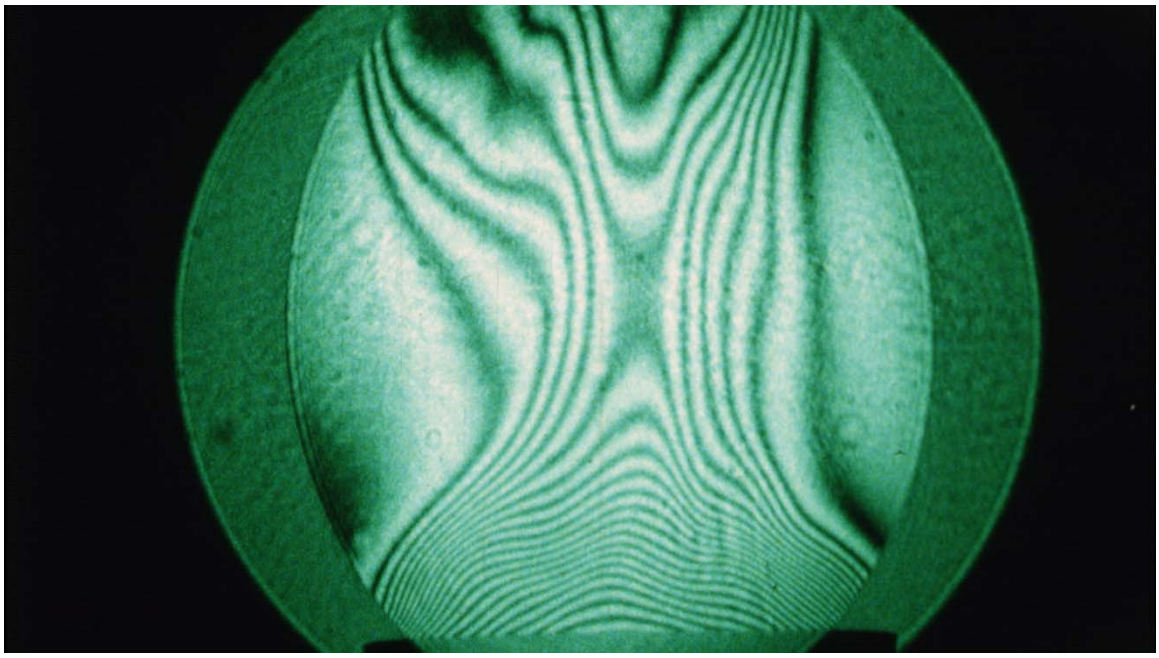


Figure 2: An image of an original interferogram.

Comparison of Holographic Interferometric Images with CFD Simulations

A quantity of interest must be specified to compare the outputs from the experiments and CFD simulations. The time-averaged fringe location is an effective quantity of interest; it is the simplest value to measure from the experiments because it requires the least amount of data manipulation. The locations of these interference fringes are easily quantified for each interferogram using image-processing techniques. The CFD simulations, however, do not output time-averaged fringe locations directly. The CFD simulation data is transformed into interference fringe orders through the use of the mixture fraction ($\gamma(x,y,z)$) and the mixture density ($\rho_m(x,y,z,t)$) at some height (x), radial location (y), length into the plume (z), and time (t). The mixture fraction and mixture density are integrated along the laser path length through the plume, the z-axis, eliminating a dimension in the CFD simulation data to resemble the two-spacial-

dimensionality of the experimental data ($\gamma(x,y)$ and $\rho_m(x,y,t)$). The mixture fractions are also multiplied by the species mass fraction to obtain mixture fractions of each species i ($\gamma_i(x,y)$) for use in the following equations.

$$\rho_m(x,y,t) = \frac{2}{3}\left[n_m(x,y,t) - 1\right]\frac{\sum_i \gamma_i(x,y)}{\sum_i \gamma_i(x,y) N_i^0} \tag{1}$$

$$n_m(x,y,t) = \frac{3}{2}\left[\frac{1}{\rho_m(x,y,t)}\right]\frac{\sum_i \gamma_i(x,y) N_i^0}{\sum_i \gamma_i(x,y)} + 1 \tag{2}$$

$$\Delta\Phi(x,y,t) = S(x,y,t)\lambda = \int\left[n_m(x,y,t) - n_0\right]dz \tag{3}$$

$$S(x,y,t) = \frac{\Delta\Phi(x,y,t)}{\lambda} \tag{4}$$

To change these quantities to interference fringes, the first step is to solve the Gladstone-Dale equation (Equation 1) for the mixture refractive index ($n_m(x,y,t)$) (Equation 2) where N_i^0 is the specific standard refraction of the stable species i [7]. The mixture refractive index is then integrated along the laser path length, the cord length (z-axis) of the plume, to determine the optical path length difference ($\Delta\Phi(x,y,t)$) (Equation 3) and then re-arranged to find the interference fringe orders ($S(x,y,t)$) (Equation 4). A data set of simulated interferograms is then constructed from the interference fringe order

data. These simulated interferograms are then processed like the experimental interferograms with a statistical analysis to calculate the time-averaged fringe positions.

Previous studies have employed a qualitative method for the validation of interferograms. CFD-simulated and experimental interferograms were evaluated visually [8, 9,10].

In each of these previous validation studies, the authors recognized that numerical data can be garnered from both the CFD-simulated and experimental interferograms. The data are transformed to refractive index or density, the experimental and CFD simulation values are plotted, and the data analyst subjectively evaluates the validation. The data analyst is the person who performs the postexperimental data analysis, distills the data into pertinent results, and compares the experiments and CFD simulations to evaluate the validation. However, the accuracy of the transformed data is highly uncertain due to the amplification of error in the transformation methods from interpolating the interference fringe orders. The Abel transform, for example, takes interference fringe orders and changes them into refractive indices; however, if the fringes do not have a somewhat linear relationship to their locations, the transform outputs a value below one that is out of the range of reality unless the system involves plasmas. In systems where the interference fringe orders relate somewhat linearly to their position, these transformation methods result in refractive indices or density values within reason [8].

These qualitative comparisons have a disadvantage. The disadvantage is that they use the subjective viewpoint of the data analyst to assess the level of validation. Typical qualitative statements for qualitative comparisons include words like “acceptable,” “excellent,” and “good” to qualify the fit of the experimental and CFD simulation data.

Even when the quantity of interest for the experiments and CFD simulations is a numerically graphed quantity, the level of validation is only as good as the data analyst's evaluation. Another disadvantage is that it is not a repeatable validation. Depending on the data analyst, different qualities of validation will be observed from different analysts and possibly from day to day with the same analyst.

A quantitative comparison would be an improvement on the current methods for validation. Recent developments in simulation science consider experiments and CFD simulations as one large Data set in which consistency can be found to quantify uncertainty through data collaboration [11]. Through this thinking, the relationship of the experiments and the CFD simulations can be quantified by a consistency analysis. Data collaboration provides a repeatable comparison that gives a quantitative measure of validation and it removes the subjectivity. To make a quantitative comparison between the experimental and simulated data, uncertainty must be computed for both to create a Data set, and a region of consistency must be found where the two quantities of interest help to explain each other’s behavior and to give a measure of consistency.

A novel comparison for interferograms is proposed in which a more quantitative approach is taken. Uncertainty is quantified for the experimental interferograms for use in a consistency analysis Data set. This comparison will result in a quantitative consistency measure for CFD simulation validation rather than the present qualitative approach.

Summary

Much of the work to study this novel comparison is set as future work since effort, which involves the participation of the simulation team, has not yet been

completed. In the next chapter, the image-processing, postprocessing analysis, and the uncertainty quantification techniques are explored for the experimental portion of this study, which is the primary objective of this thesis. The final chapter shows the results of these analyses, along with a discussion of the future work where the quantitative comparison will be performed through a consistency analysis on the Data set.

METHODS

Experiment and Data Set Setup

Four holographic interferometry experiments of a laminar helium plume were performed on July 10, 1984 at the University of Stuttgart, Germany. The interference fringe locations within the plume were measured in the four experiments at helium flow rates between 2 and $2.5 \cdot 10^{-4}$ m^3/sec. The known experimental prior uncertainty is given in Table 1. Due to the fact that no local measurements of temperature and pressure were made during the experiments, the ranges were garnered from local weather data [13].

To create the helium plume interferograms, a 2-watt continuous-wave argon laser beam at a wavelength of 514.5 nm was split into expanded object and reference beams [14]. The object beam was reflected through the helium plume, while the reference beam was reflected through quiescent air. The two beams were then superimposed on a holographic plate. Through destructive and constructive interference of the two beams, an interferogram was created. A high speed camera captured a continuous movie of the interferograms at 1000 frames per second for approximately 1.3 seconds for each condition. About fifteen years later, the data set films were digitized for analysis by IWF - Wissen und Medien gGmbH at a resolution of 1980 by 1080 pixels.

Table 1: Experimental conditions

Experimental Parameter	Parameter uncertainty range
Temperature	22-32 °C
Pressure	96.68-101.5 kPa
Wavelength	514.5 ± 50 nm
Helium Composition	99.995-100%

Determination of the Characteristic Data Subset

Each data set of interferograms was composed of approximately 1300 interferograms corresponding to ten puff cycles. Processing an entire data set was prohibitively time consuming; therefore, a characteristic data subset was determined to accurately portray the overall data set in terms of fringe location. This analysis revealed that a characteristic data set of five puff cycles in which every fifth interferogram was processed accurately reflected the fringe locations of the full data set with an average difference in fringe position around 1% of total radius, with a maximum difference of 2.75% of total radius.

Determination of the Statistical Data Subset

A statistical analysis was performed on the processed characteristic data subsets to determine average interference fringe locations. To determine these mean values, a statistical data set of two or more data points was required for the development of statistical moments (i.e., mean and standard deviation) as well as the probability density functions. A total of eight possible statistical data subsets were processed from the first and fourth experiments due to their similar helium flow rates (i.e., 2.25 (Helium_01) and $2.3 \cdot 10^{-4} m^3/sec$ (Helium_04)). The interactive nature of the plume's boundary layer

prohibited the independent evaluation of both the left- and right-hand sides of the interferograms, resulting in a division of possible statistical data subsets by a factor of two. This meant that only four were able to be used as the statistical data subsets, as shown in Figure 3.

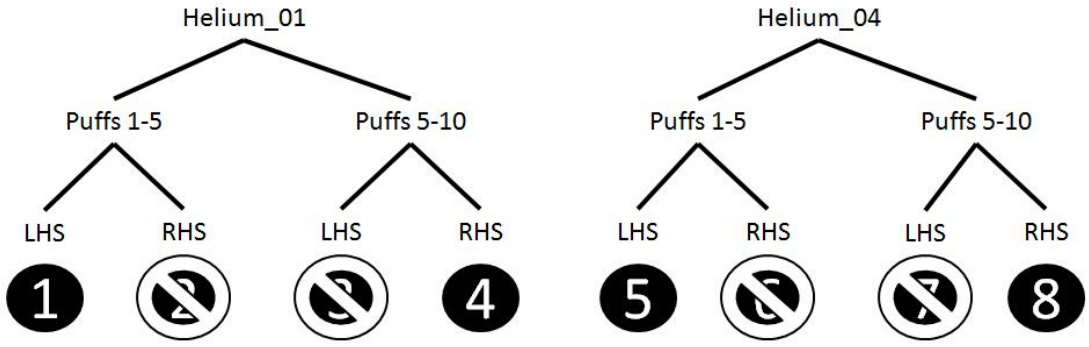


Figure 3: Statistical data subset selection

IMAGE-PROCESSING

Every image in the four statistical data subsets was processed through the following image-processing steps. A MATLAB code for each major step in the image-processing method, except resolving, was created; the five steps were (1) brighten, (2) register, (3) crop and label, (4) resolve, and (5) compile. Each step refined and filtered the data until the helium fringes were isolated at the specified sampling heights and were written to a spreadsheet file for statistical postprocessing.

Brightening

The brightening step normalized the light intensity histograms of the interferograms. This step was very important because the differences in intensity inherent in the data sets resulted in differing fringe widths and image noise, and thus differing fringe locations. The first step in the brightening process was to change the images from color to gray-scale. This change was done by the rgb2gray function in MATLAB where the red, green, and blue values were weighted to produce differing intensities from black (zero) to white (one). The equation for this conversion in the MATLAB function is shown in Equation 5 where the colors of red, green, and blue correspond to the color components for a range from 0 to 256. The output gray scale values also have a range of gray values from 0 (black) to 256 (white)

$$\text{Gray level} = 0.2989 \cdot \text{Red} + 0.5870 \cdot \text{Green} + 0.1140 \cdot \text{Blue} \quad (5)$$

The brightening code then determined the histogram of the gray-scaled images by counting the instances of pixels with a gray color quantity specific to a value between 0 (black) and 256 (white). An example histogram is shown in Figure 4.

The histogram for each image in the data set was then averaged by a summation of all the data set histograms and divided by the total number of images for the data sub-set. This averaged histogram was then applied to each image for intensity normalization. This normalization was done by the function histeq. This function determined a gray-scale transformation through minimization to alter the image to fit the input histogram. This minimization was done by reducing the difference between the chosen gray-scale transformation histogram and the image histogram. The gray-scale transformation is a map for the function; it specifies where the gray levels should be altered to fit the input histogram. With the brightening process complete, the images looked like the example shown in Figure 5. These brightened images were kept as gray-scale images for future input into MATLAB functions that required gray-scale image inputs.

Registration

Once brightened, the images were ready for the next major step in the image processing method. Registration as the next major step had two roles; one role was to sharpen the images, and the other was to reduce the image shifting by registering the images to three specified points in the first image of the data set. The images were enhanced using the sharpening filter in the fspecial function to be applied later in the

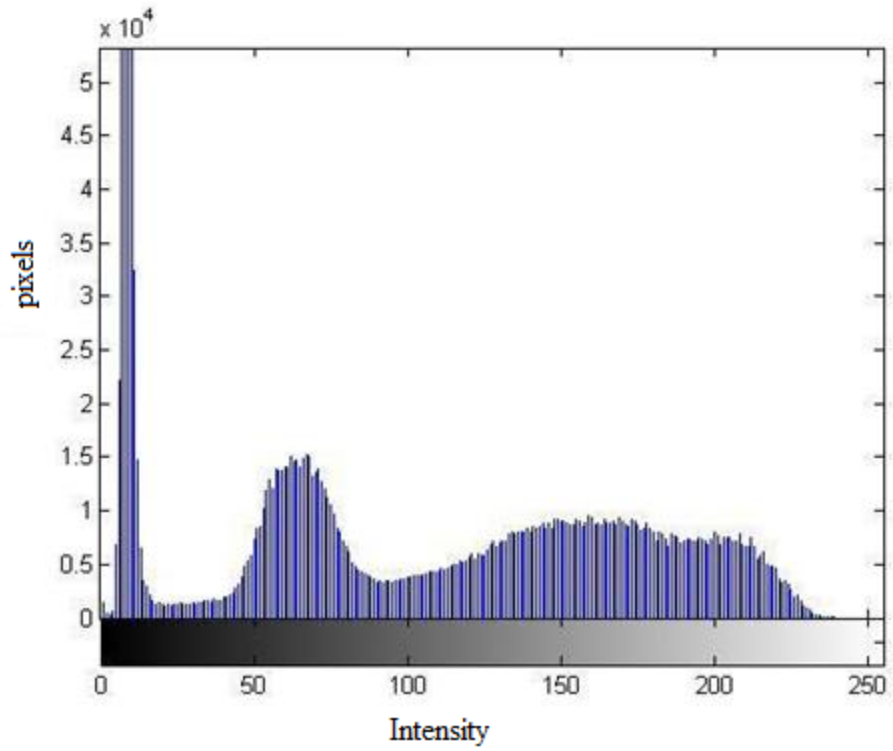


Figure 4: Histogram of an image before brightening.

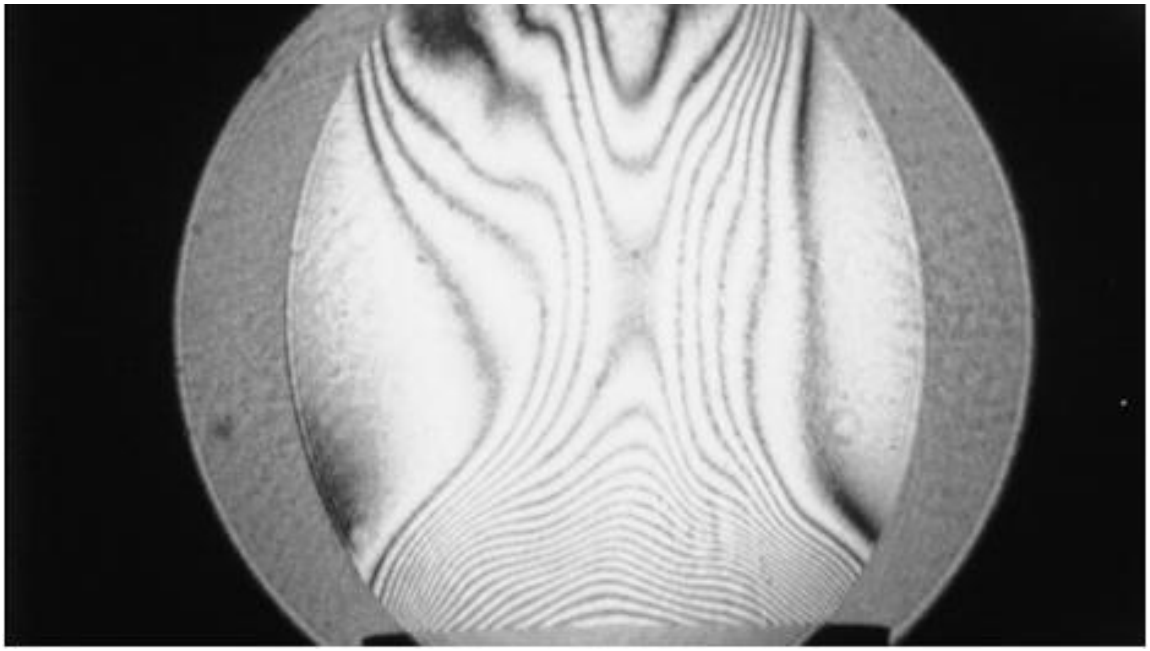


Figure 5: Example of a brightened image from Helium_01

imfilter function. This sharpening increased the accuracy of the registration by increasing the contrast. The sharpening filter in the fspecial function filters an image by subtracting a blurred version of the image from the original. This sharpening was performed using a negative of the Laplacian filter with a specified parameter called alpha. The default value of 0.2 was used for this parameter. The equation for this value is shown in Equation 6. The equation is only dependent on the value of alpha specified and not the image to be filtered. The value of alpha controls the shape of the Laplacian filter.

$$\frac{1}{(\alpha+1)}\begin{bmatrix} -\alpha & \alpha-1 & -\alpha \\ \alpha-1 & \alpha+5 & \alpha-1 \\ -\alpha & \alpha-1 & -\alpha \end{bmatrix} \tag{6}$$

The filter from the fspecial function was then input with the image to be sharpened in the imfilter function. The imfilter function used double-precision floating point arithmetic to determine the output for each pixel in the images. The option of replicate was specified in the imfilter function to set any output value that exceeded the bounds of the maximum for the image type as being equal to the closest image border value. In other words, for a pixel in an image with an output value larger than 255 or smaller than 0, the pixel was given a value that corresponded to the closest image border value. Once filtered with the inverse Laplace filter, the images were more sharp and clear for the rest of the registration process. An example of an image after sharpening is shown in Figure 6.

In registering the images, a set of input points were correlated from a data set image with a set of base points from the primary image. Each image of the data set was

Figure 6: Example of an image after the sharpening filter has been applied.

correlated by the two points found at the corners of the pan and a third stationary point that was constant through the data set. The registration was done through a Matlab function called cpselect where the user manually selected the input points and base points for the data images to register them. The function brought up a user interface like that shown in Figure 7 where the three input points are specified for each image. The third stationary point for registration was different for each data set and thus had to be proved to be constant through the entire set before the registration process could be started.

Using the cpcorr function, the input points were used to orient the data images in-line with the primary image without stretching it. A 11-by-11 matrix around the input point was extracted as well as a 21-by-21 matrix around the base point. The normalized cross-correlation of the matrices was then performed and a peak value of correlation was observed. This peak was then used to adjust the data image to the primary in the imtransform function with the spatial transformation specified by the cp2tform function with the nonreflective similarity transformation type. This imtransform function only

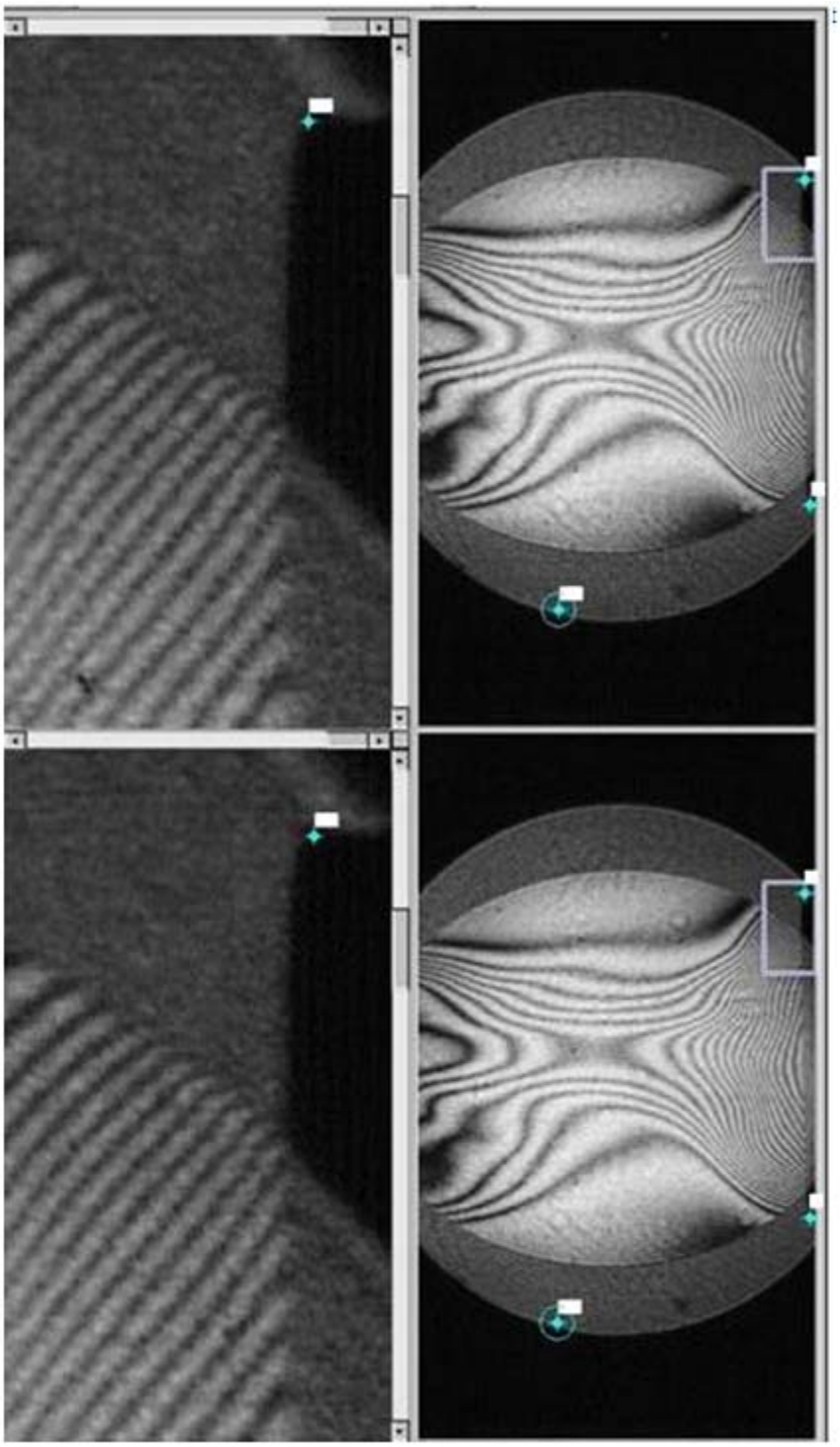


Figure 7: Input points for registration of an image to the primary image of the data set

moved the images a maximum of four pixels in one direction from their previous orientations. An adjustment up or down, left or right to re-orient the data image to the primary image axis was all that was performed using this specified spatial transformation type. Once aligned, the images were registered and ready for the next processing step.

Cropping and Labeling

Once brightened, sharpened, and registered, the images were filtered again and then cropped to a smaller size. The second filtering was performed to enhance the transformation of the image from gray-scale to black and white. To analyze the fringe data, the images had to be in only black and white to correctly label the fringes in the connected component analysis. Five filters in total were applied to the images to reduce noise, speckling, and fringe intersections. The first filter, adapthisteq, split the image up into a default number of 64 tiles (an eight-by-eight sectioning of the image) to enhance their contrast using the histogram of each tile. The boundary pixels of each tile were bi-linearly interpolated to blend them together with other boundary tiles; a contrast limit of one would result in the maximum contrast in each tile possible and a limit of 0 would result in no change in the contrast. This filter increased the contrast between the fringes and the spaces between them. An example of a data image with this filter applied is shown in Figure 8.

The images were then changed from gray-scale to binary color, black and white, by the im2bw function based on the luminance threshold value specified. For any pixel that exceeded the luminance threshold, it was labeled white with a value of one. For any that fell below the threshold, they were labeled black with a value of zero. The luminance

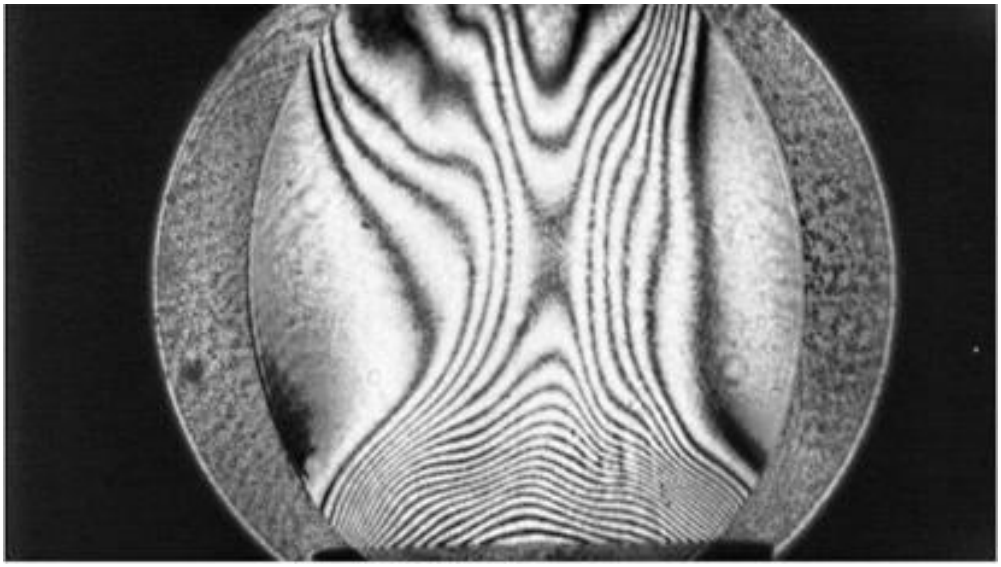


Figure 8: Data image after the adapthisteq filter.

threshold varied between data sets due to the differences in light intensity, thus different values were used as the threshold for different data sets. An example of an image with the im2bw function applied is seen in Figure 9.

Once changed to a black and white image, a bevy of filters specific to binary images were applied. These filters reduced noise, eliminated speckles, and separated fringe intersections. The filters were all options of the bwmorph function. The clean filter removed any pixels with values of one if they were surrounded by values of zero. The fill filter changed values of zero to ones if they were surrounded by values of one. The majority filter set a pixel originally at one to a value of zero unless the majority (five or more in the surrounding eight pixels) were also one. With the thin filter specified in the bwmorph function, large sections of ones in the image would have one pixel from each side turned to zeros. An example image where this bwmorph sequence was performed is shown in Figure 10.

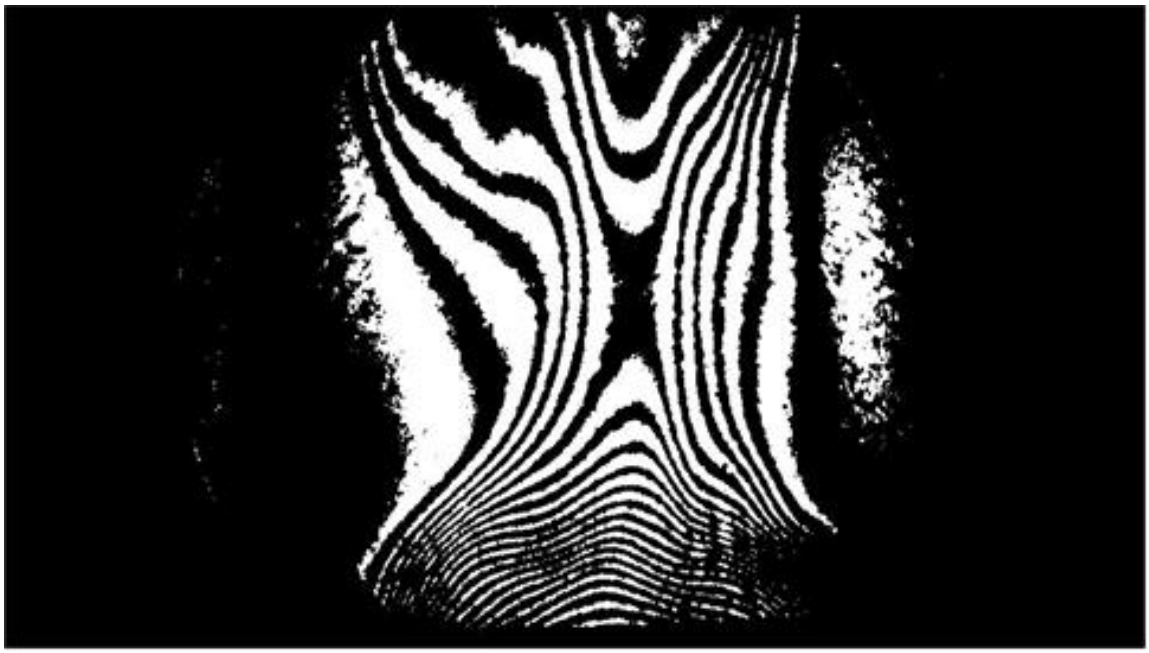


Figure 9: An example of an image changed from gray-scale to black and white.

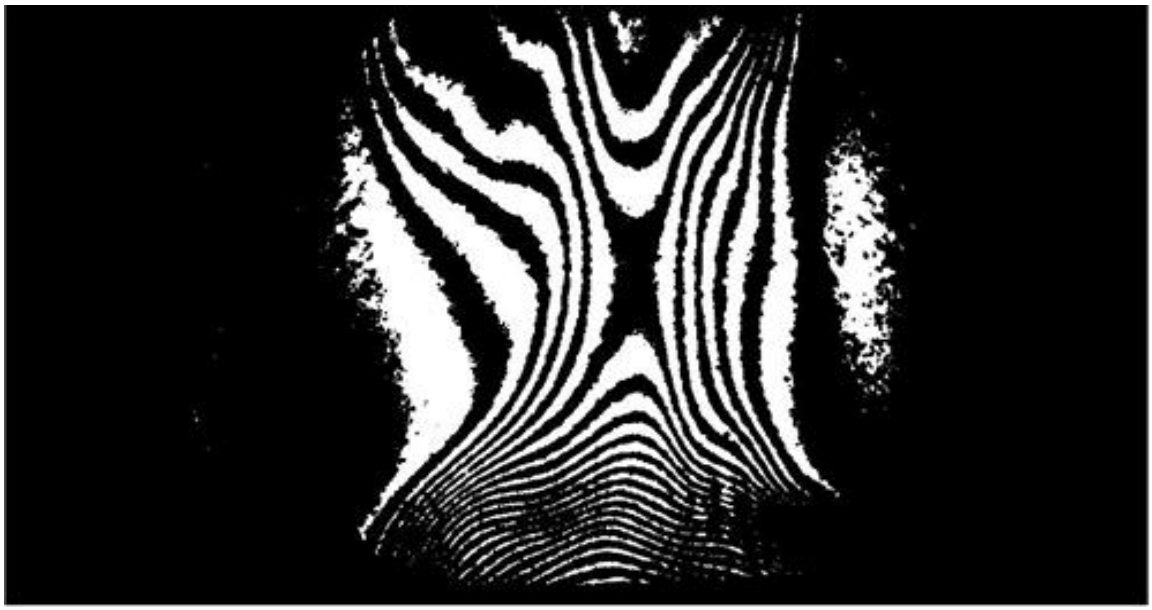


Figure 10: Example of an image filtered by the bwmorph function.

Another filter specifically for binary images in the medfilt2 function reduced the speckle noise and filled in the speckles with the majority designation surrounding it. The matrix used for the determination of majority was set as the default eight-by-eight. In comparison with the bwmorph filters, medfilt2 produced a much more noticeable change in the image clarity. An example of an image filtered using the medfilt2 function is shown in Figure 11.

Due to the nature of the helium plumes and interference fringes being created by waves canceling out, fringes were black in the original data images. For further processing, the fringes had to be turned white so they could be labeled in the connected component analysis. With the function imcomplement the image's black (zero) areas were relabeled as white (one) and the white (one) areas were relabeled as black (zero). An example image with the imcomplement function applied is shown in Figure 12.

Figure 11: An example of an image filtered by the medfilt 2 function.

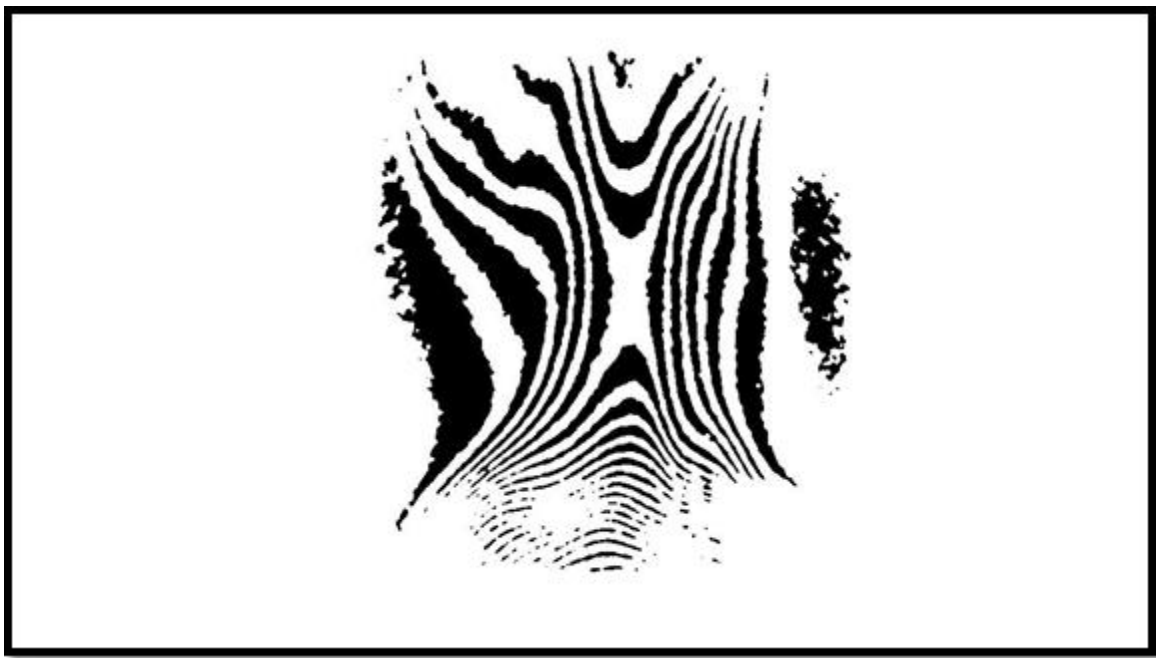


Figure 12: An example of an image with the imcomplement function applied.

Next, the image was cropped to the axial heights above the port rim of x = 3 cm and x = 5 cm. The rectangular crop of the image was 926 pixels (Δ x = 5.2 cm) wide to encompass the entire plume and 356 pixels (Δy = 2 cm) tall to reach the two measurement heights. The image pixels were numbered from the top left hand corner of the image with the x-axis as the horizontal and the y-axis as the vertical. Figure 13 shows the cropped area with measured pixel distances. An example of a cropped image is shown in Figure 14.

The next step in the cropping code was to label the interference fringes. The bwlabeln function determined the connected components (large sections of similarly labeled pixels) in the binary images and then labeled them with increasing numbers from one to infinity. Once labeled numerically, a color map showed each connected

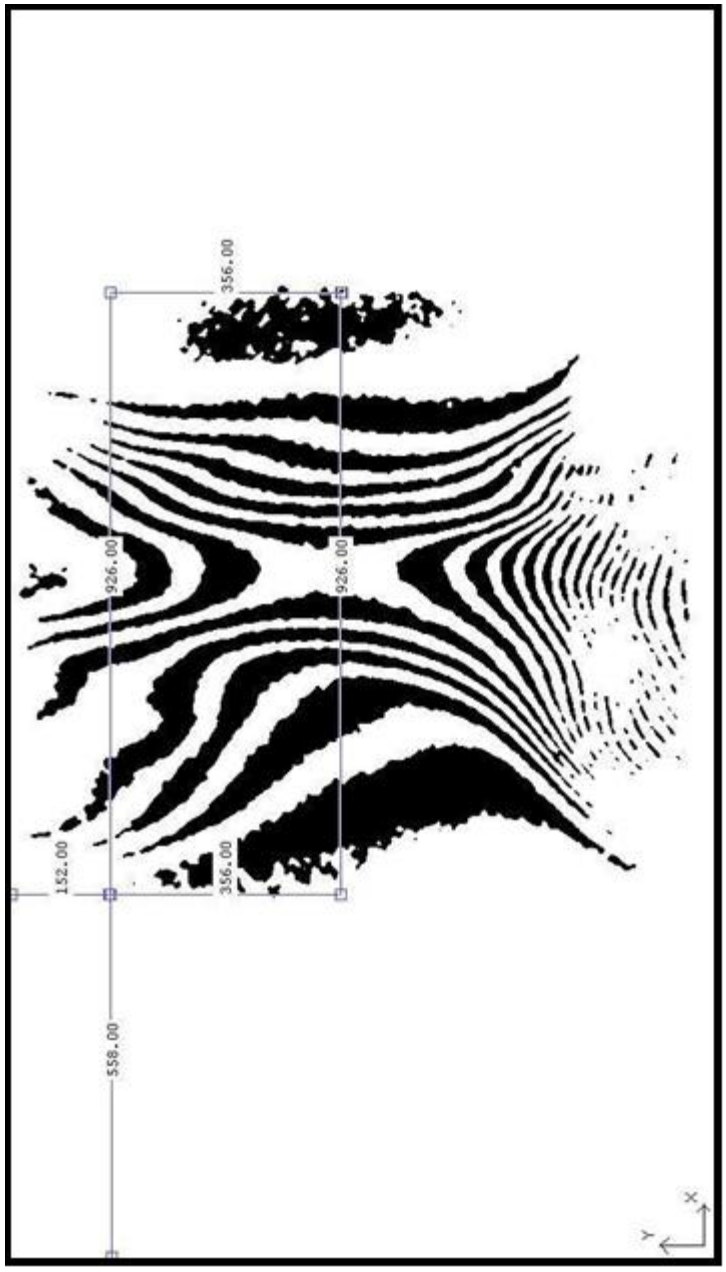


Figure 13: Pixel measurements and positions for the cropping height in image-processing.

Figure 14: An example of a cropped data image.

component by using the label2rgb function. An example of a labeled and colored image is shown as Figure 15.

Resolving

Before the fringe's locations could be compiled, the data images had to be filtered by hand to resolve the fringes and eliminate noise. This processing step was performed by opening each of the images in Photoshop and deleting pixels containing noise or that connected fringes. An example of a resolved image is shown as Figure 16.

Compiling

Once resolved, the images contained only the fringes ready for the final sampling and compilation. The final sampling was performed by cropping the image using the imcrop function at x = 3 cm and x = 5 cm (top and bottom of the image) in one pixel height segments. For x = 5 cm, the cropping started at the origin (top left hand corner) of the image. For the x = 3 cm, the cropping started at the bottom left of the image at pixel

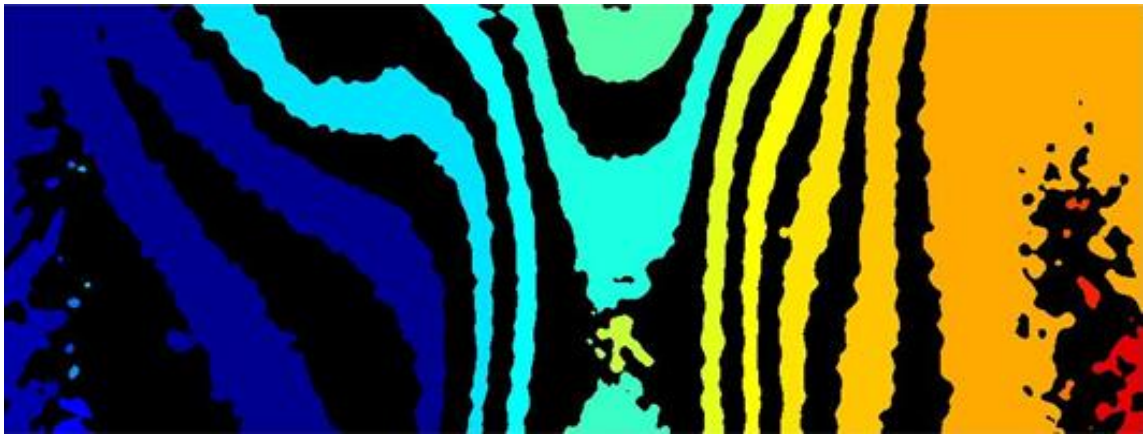


Figure 15: An example of a data image that has been labeled and colored.

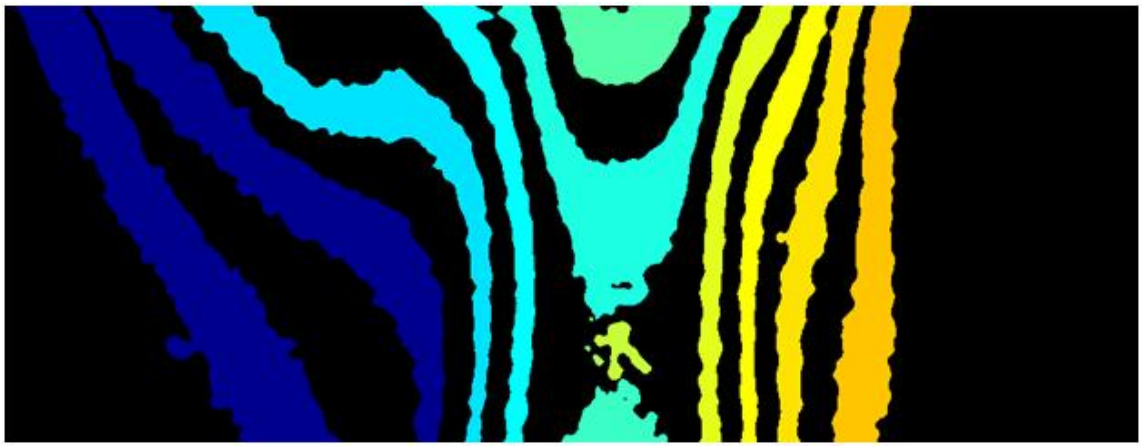


Figure 16: An example of a resolved image.

height 357 (x = 2 cm) with a crop height of one pixel. Examples of samples from each of these heights are shown in Figure17 and Figure 18.

These final samplings were labeled with the bwlabeln function. These numerical labels were then transformed into interference fringe order values for analysis. An example of the interference fringe order numbering is shown in Figure 19.

For accurate characterization of the fringes in the final analyses, the fringes had to be skeletonized, stripped of pixels equally on each side, until the innermost pixel of each fringe was isolated. This was performed using the regionprops function with the specified option centroids. This function analyzed the image areas to determine the center of their masses and then compiled them into a vector. The values given in the centroid output

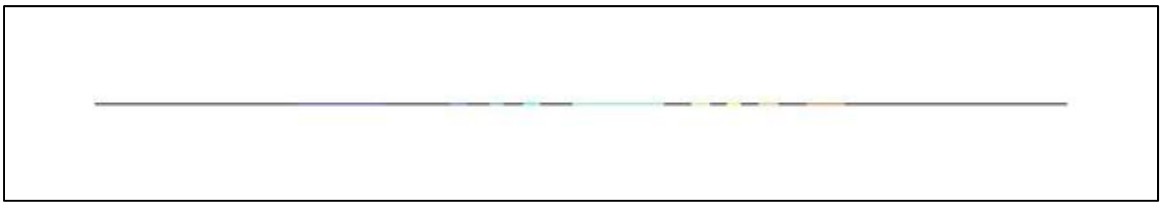


Figure 17: A one pixel height sampling at x = 3 cm.

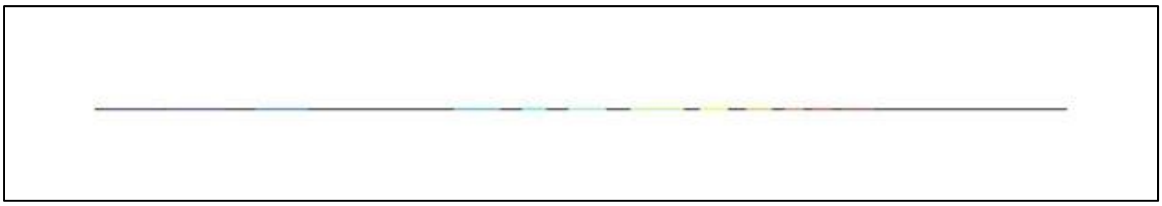


Figure 18: A one pixel height sampling at x = 5 cm.

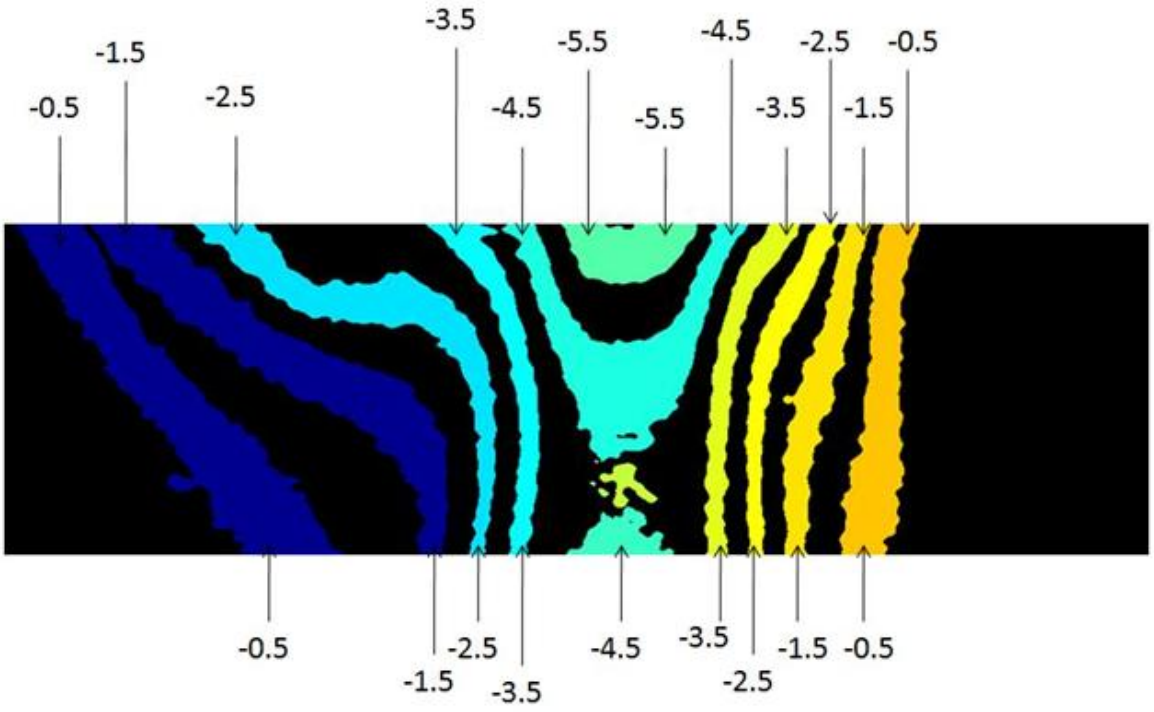


Figure 19: Interference fringe ordering

vector were then written to a new matrix with the corresponding fringe order and radial location. Fringe orders are negative half values, numbered from the outside of the plume to the inside due to the optical path length difference where there is destructive interference and that the refractive index of helium is less than that of air. Once numbered and skeletonized, the images were written to a spreadsheet file for compilation. The function dlmwrite took the renumbered image matrix and transferred it to a spreadsheet file.

POSTPROCESSING ANALYSIS

The spreadsheet files were then read into MATLAB matrices for the post-processing analysis. For each fringe at each measurement height, a probability density function (PDF) was developed. The probability density function showed the fringe positions over time; a high probability was attached to regions where the fringe remained for a substantial portion of the data set. Probability was determined by counting up all the occurrences of a fringe, the total number of fringe occurrences in a range of radial location, then dividing the two. Each bin is shown as a vertical bar on the PDF graph. From these PDFs, the average fringe location (y_{avg}) (Equation 7) was determined as well as d, the standard deviation (Equation 8) where y_i is the radial location and PDF(y_i) is the probability for that fringe at that radial location.

$$y_{avg} = \sum_{i=1}^{n} y_i \, PDF(y_i) \tag{7}$$

$$d = \sqrt{\sum_{i=1}^{n} \left(y_i - \bar{y}\right)^2 \, PDF(y_i)} \tag{8}$$

UNCERTAINTY QUANTIFICATION

Three types of experimental uncertainty effects were quantified in reference to the average fringe location: measurement uncertainty from random data variation, scenario uncertainty from bias and parameter uncertainty, and model uncertainty from image-processing. The measurement uncertainty, random experimental data variation, was due to small, random changes in experimental conditions like flow rate, temperature, pressure, and wavelength that varied the position of the interference fringes or the number of fringes observed. The measurement uncertainty was quantified using a statistical analysis where an average and standard deviation was determined using a PDF for each fringe.

Scenario uncertainty is any source of uncertainty due to how the experiments were performed like weather conditions and measurement methods. Bias uncertainty comes from a repeated measurement that consistently gives values higher or lower than the actual. The bias uncertainty was determined by comparing the experimental measurements at x = 0 (top of the port rim) to an analytical solution where the plume was pure helium. At x = 0, the helium just entered the system and had not mixed with air, meaning that the refractive index gradient was known and the interference fringe orders were computed using a simplified interference fringe order equation (Equation 9) [15].

$$S(0,y) = \left[n_{helium} - n_{air}\right]\frac{A(y)}{\lambda} \tag{9}$$

The interference fringe order equation therefore had an analytical solution, shown in Equation 9, where refractive index of helium (n_{helium}), refractive index of air (n_{air}), cord length (A) at radial location (y), and wavelength (λ) give the interference fringe order S(x,y) at height x = 0 and radial location (y). The variables, equation, and geometry for the cord length are shown in Figure 20. The parameter uncertainty was also addressed in the analytical solution through changing the refractive index of the air and helium with respects to temperature, pressure, laser wavelength, and composition helium.

Model uncertainty in the image-processing was a result of subjective quantification of code parameters that effected interference fringe width. The model uncertainty was determined using a Box-Behnkin sensitivity analysis (Figure 21) where the ranges of the three most influential image-processing code parameters were varied to output their effects on fringe location. The Box-Behnkin design used the ranges of the three parameters to define thirteen different input cases. Each input parameter was assigned an axis and the minimum and maximum of the given ranges were assigned as shown in Figure 21.

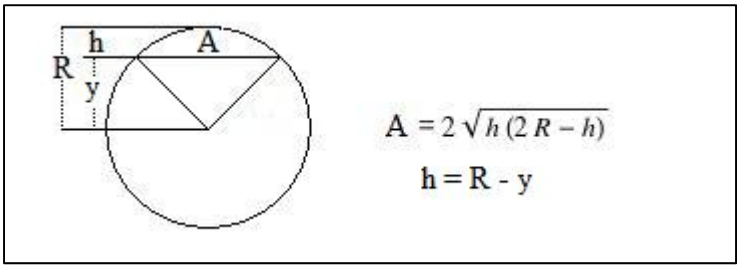


Figure 20: Cord length geometry and corresponding equation

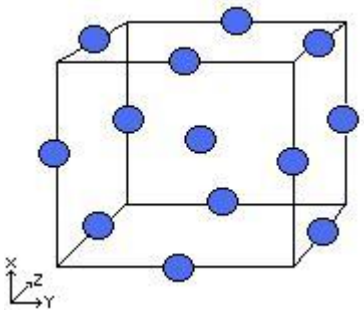


Figure 21: Box-Behnkin design

RESULTS AND DISCUSSION

Experiment

After the images were processed for the data subsets using the image-processing techniques, a statistical analysis was performed and PDFs were computed to obtain the quantity of interest and to determine the measurement uncertainty. An example of a PDF is shown in Figure 22 for fringe order -0.5 at height, x = 3 cm. A PDF was made for each fringe order separately to determine their individual time-averaged locations. The PDFs for all the fringes at each measurement height are shown in Appendix C.

Most of the interference fringe PDFs have a bimodal-like shape due to the puffing of the plume. The fringes usually have two highly probable radial locations during the puff cycle due to the location of the gradients moving outward radially during puffs and inward during pinches, the place between two puffs. Certain fringes that are only visible during a puff have PDFs without bimodal-like shapes. These fringes are seen in only a small range of radial location at the center-line and have little location variation. These fringes that are only visible in a puff have few occurrences. Their average location and deviation are affected by the decreased sample size. Interference fringes -4.5 to -10.5 have smaller sample sizes than fringes -0.5 through -3.5. As seen in Figure 23, the fringes that occur only in the puff (interference fringes -4.5 to -10.5) occur less frequently than the fringes that are present in the puff and also during the pinch between puffs (interference fringes -0.5 to -3.5). This reduction in sample size affects the

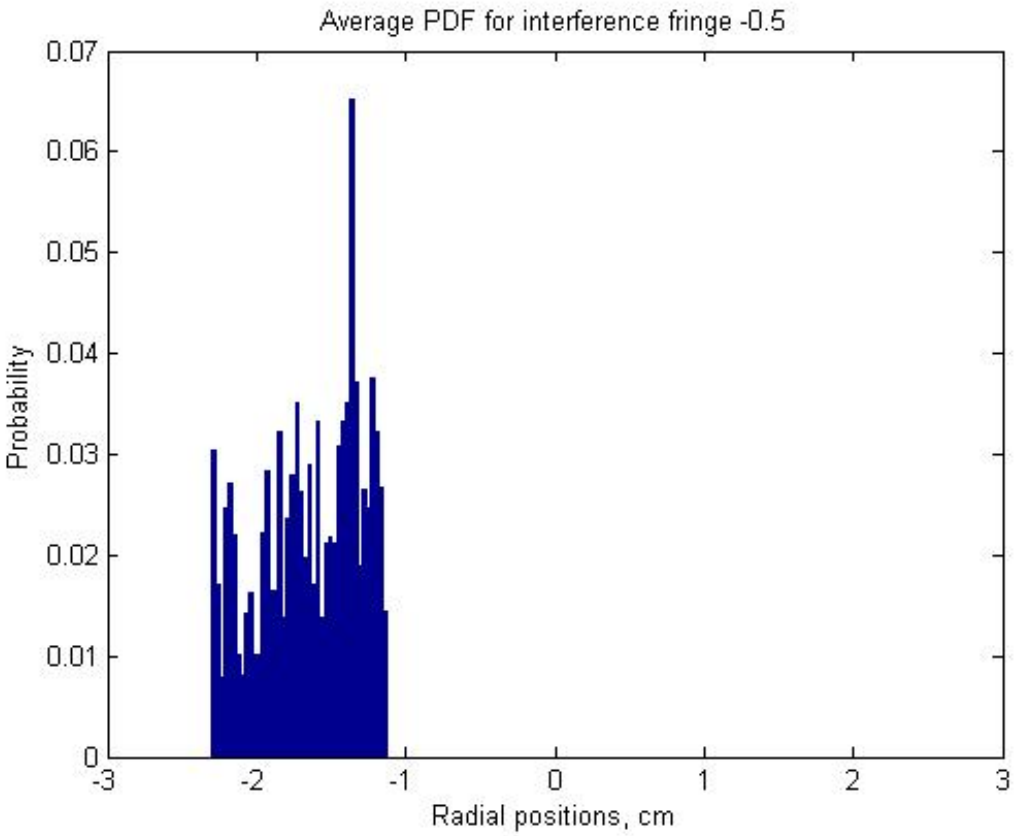


Figure 22: The PDF of fringe -0.5.

accuracy of the time-average and standard deviation calculations for these fringes. In the future, statistical methods for determining the average and deviation for small sample sizes should be used to increase the accuracy of the measurement uncertainty for interference fringes -4.5 to -10.5.

The first moment of the fringe location PDF gave the time-averaged fringe location. A standard deviation was also computed from each PDF through the second moment to define the uncertainty due to random data variation, shown in Figures 25. The maximum uncertainty due to measurement uncertainty was calculated to be 5.4 millimeters or 23.4% of total radius.

The scenario uncertainty was computed next. Average fringe profiles at x = 0

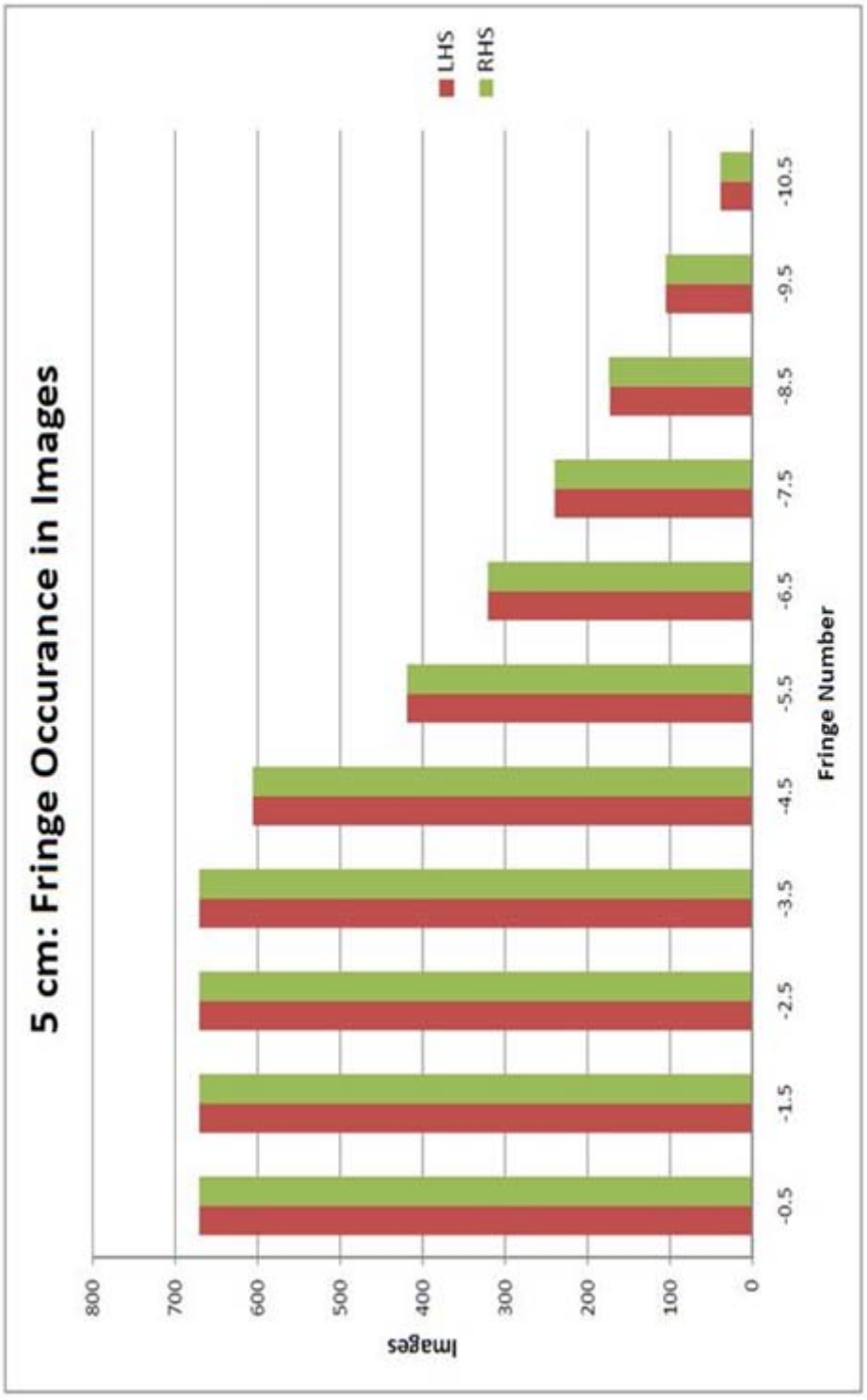
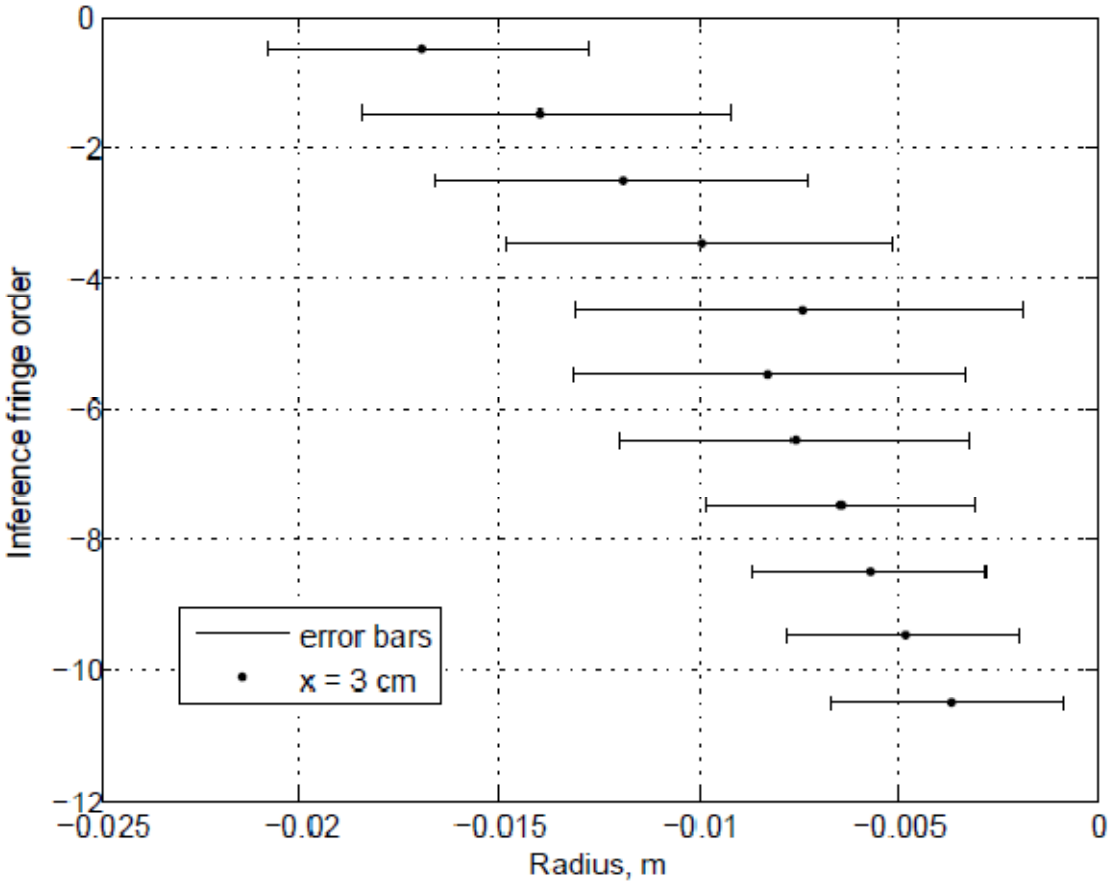
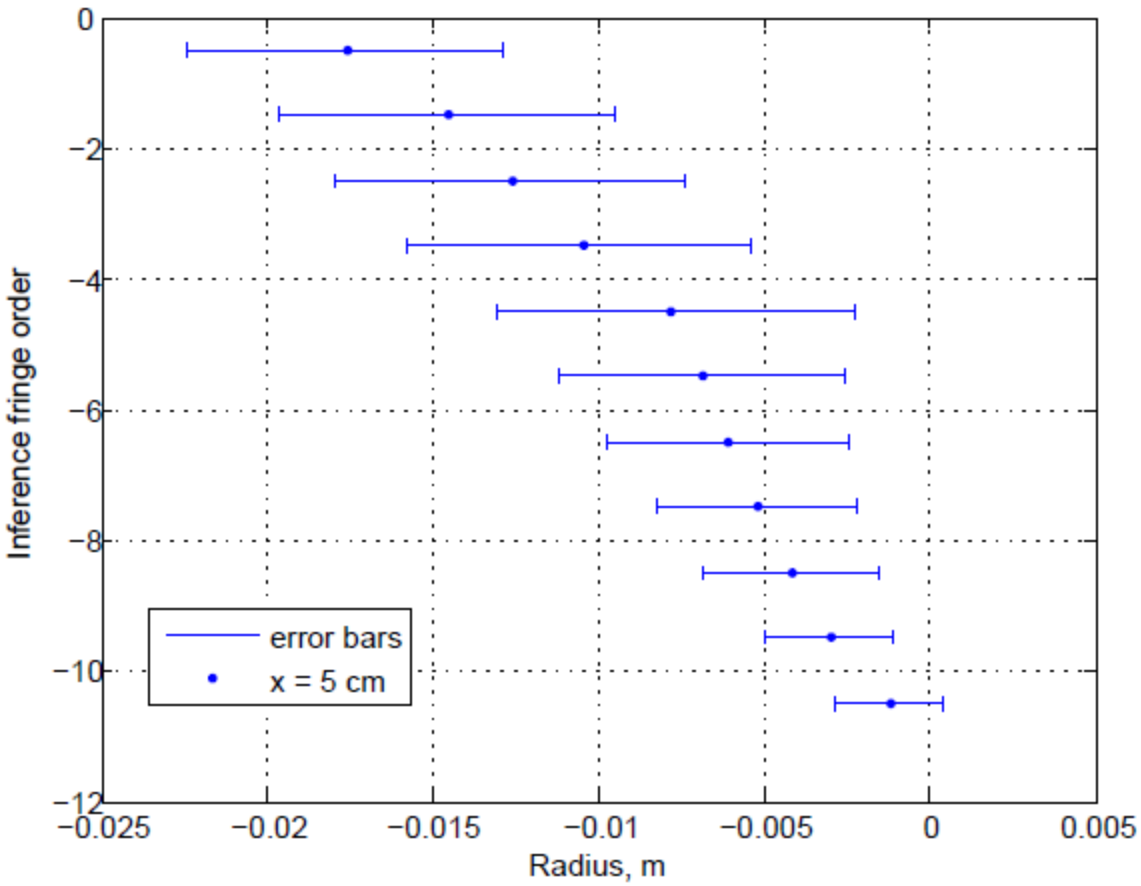


Figure 23: Total number of images with fringe orders at $x = 5$ cm



(a)

Figure 24: Time-averaged fringe profiles. (a) x = 3 cm, (b) x = 5 cm



(b)

Figure 25: continued.

were developed for all four data subsets and graphed with the analytical solution (Equation 9) shown in Figure 25 along with the range of parameter uncertainty for the analytical solution. To quantify the parameter uncertainty range of the analytical solution, the temperature, pressure, composition of helium, and wavelength (shown in Table 1) were set at their limits and the analytical solution (Equation 9) was computed. Two lines exist for the analytical solution to define this region of uncertainty. The higher analytical line defined the analytical solution at the lowest pressure, highest temperature, highest wavelength, and lowest helium purity. The lower analytical line defined the analytical solution at the highest pressure, lowest temperature, lowest wavelength, and highest helium purity.

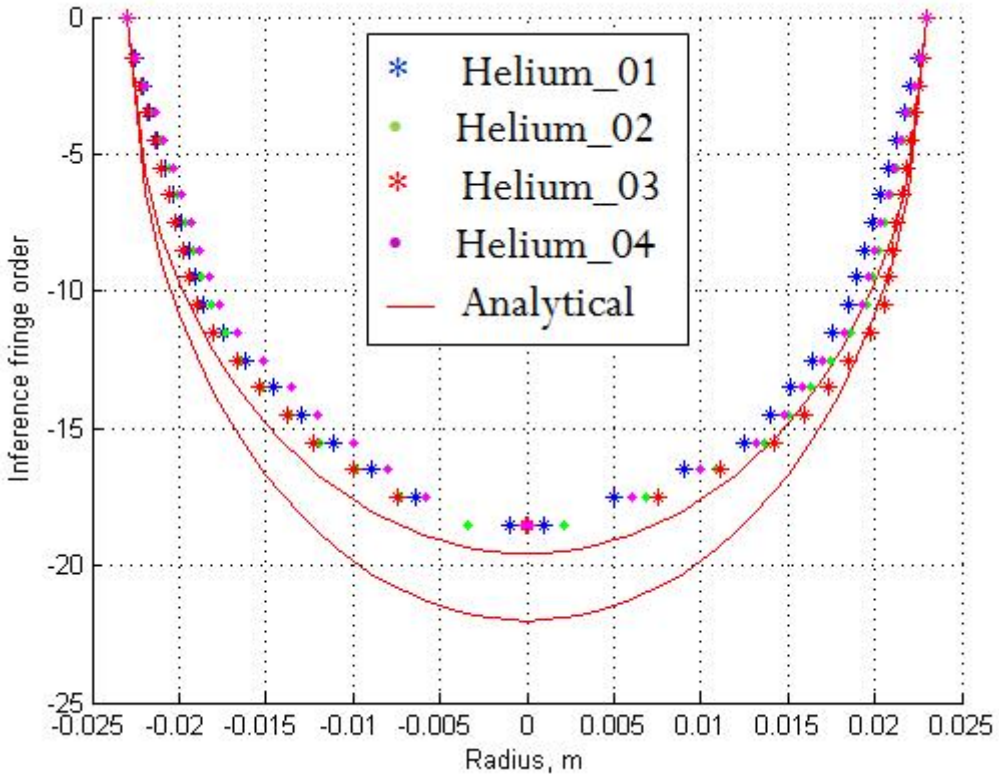


Figure 25: Fringe profile at x = 0 for all the data sets

An average fringe location was determined from the processed images and a standard deviation was computed from the data points in Figure 25. These data were then interpolated to convert the uncertainty from radial location (horizontal bars) to interference fringe order (vertical deviation) using MATLAB interpolation functions. The interpolation was performed by taking the edges of the standard deviation bars in Figure 26 interpolating them to make a continuous line above and below the experimental measurements, and then the corresponding locations of the measurements were pulled out of the interpolated lines creating vertical uncertainty ranges (Figure 27). The maximum uncertainty due to scenario uncertainty is -3.7125 fringes or 35.4% of total number of fringes. The scenario uncertainty was the largest source of uncertainty in this system.

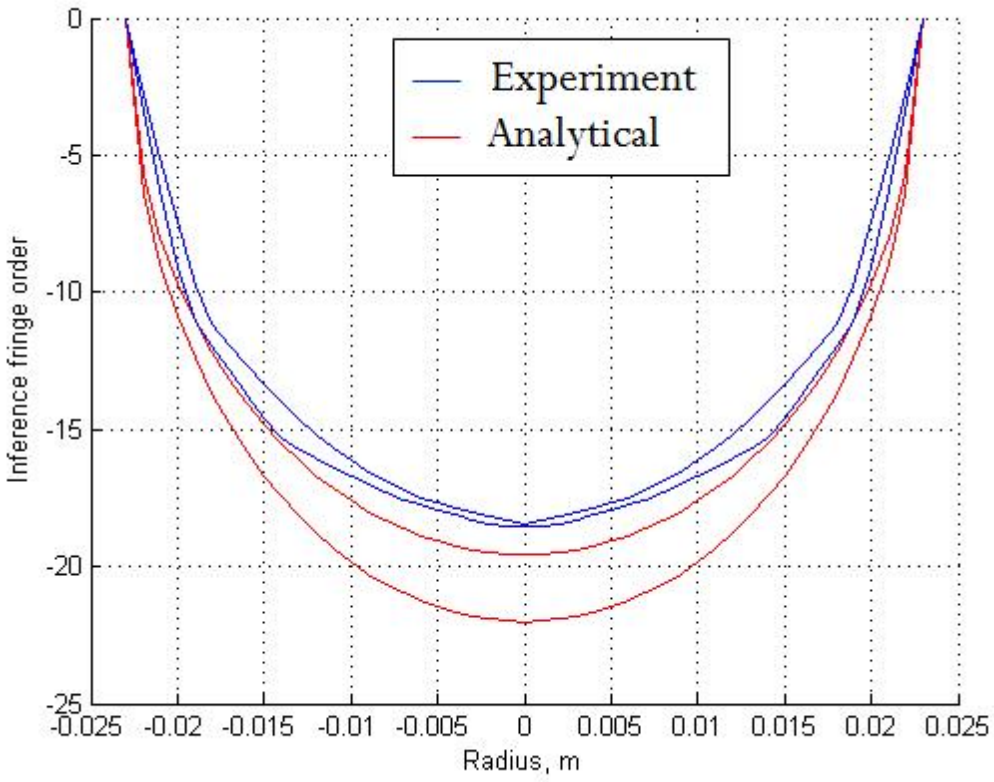


Figure 26: Interpolated bars for horizontal to vertical uncertainty

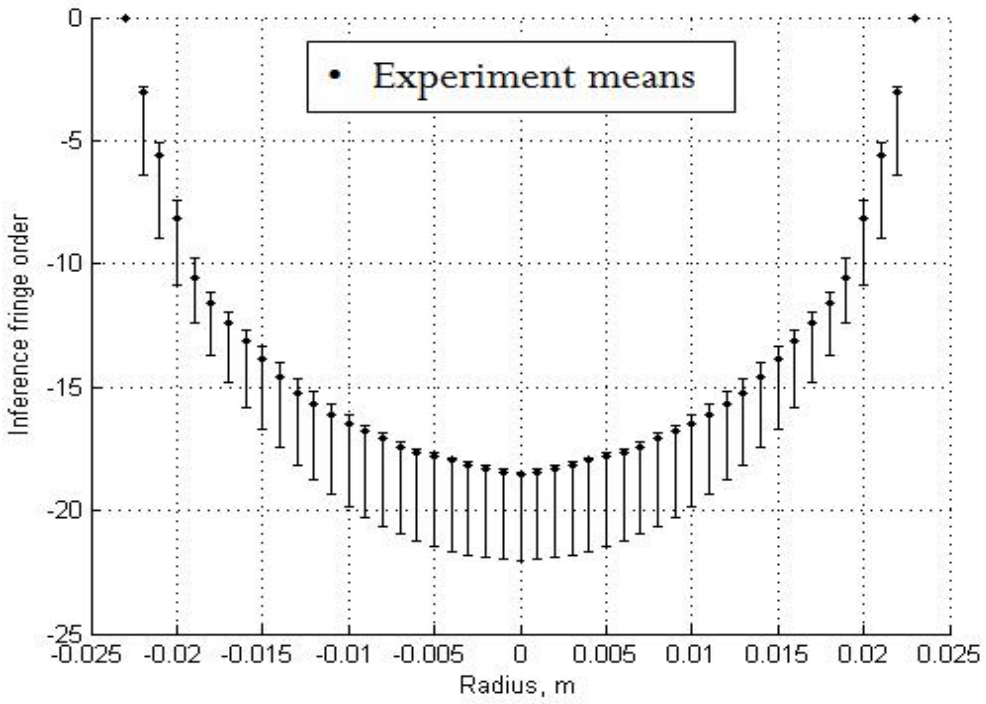


Figure 27: Bias error and experimental parameter uncertainty at x = 0

A sensitivity analysis was then performed to calculate the model uncertainty due to image processing at a height of x = 3 and 5 cm. The three parameters that had a significant effect on fringe location were the threshold for the image to black and white function (im2bw), the registration of the image, and the adaptive histogram equalization contrast limit filter in the adapthisteq function. A Box-Behnkin design was employed to determine the sensitivity of the fringe locations to the given parameters for both Helium_01 and Helium_04 separately. Two different sensitivity analyses had to be performed due to the difference in light intensity in the data sets which resulted in different ranges of contrast limit for each data set. The Box-Behnkin design used for Helium_01 is shown in Table 2. A maximum uncertainty of 34 pixels or 0.955 millimeters was observed, 4.15% of the total radius. It was theorized that this value was

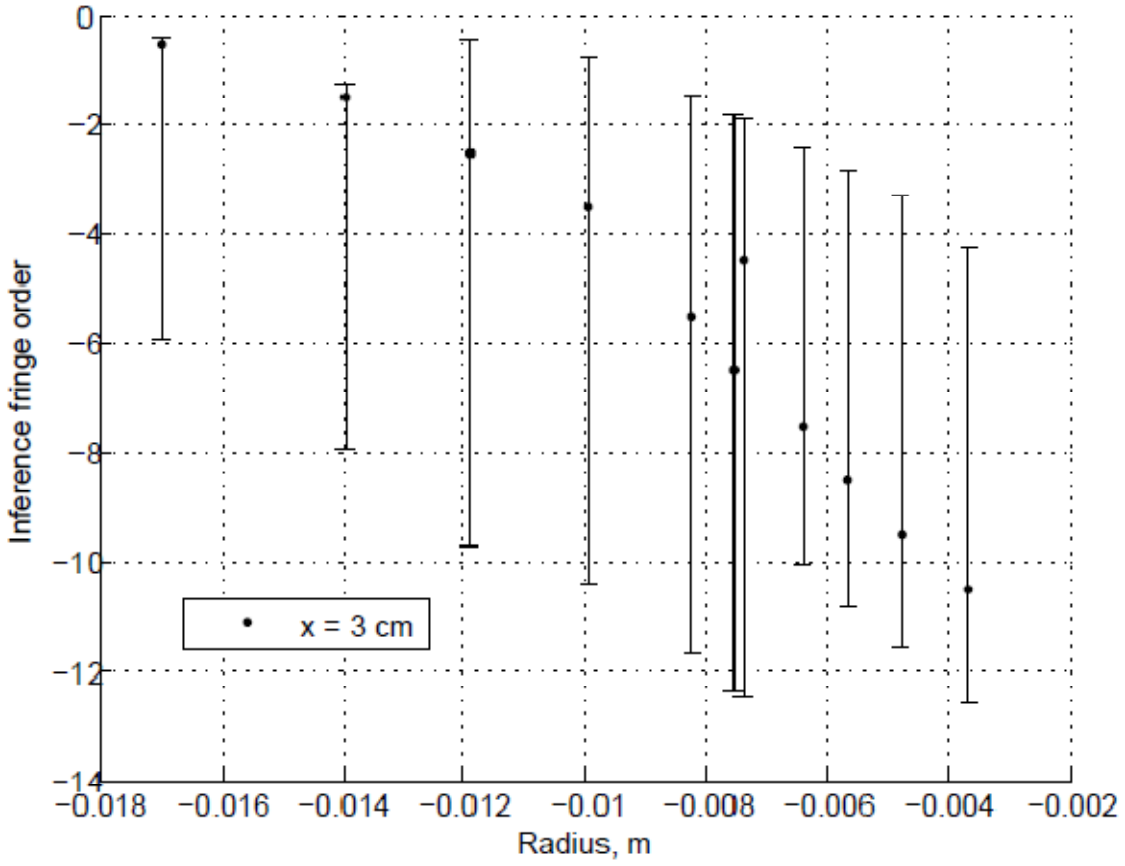
Table 2: Example of a Box-Behnkin simulation design

Parameter	Lower bound	Middle	Upper bound
Im2bw threshold	0.80	0.81	0.82
Registration	-3	0	3
adapthisteq contrast limit	0.01	0.135	0.17

lower than expected because the uncertainty in the registration of the image was not adequately characterized. In future, other methods of quantifying the registration uncertainty should be used to more accurately characterize its affect on fringe location.

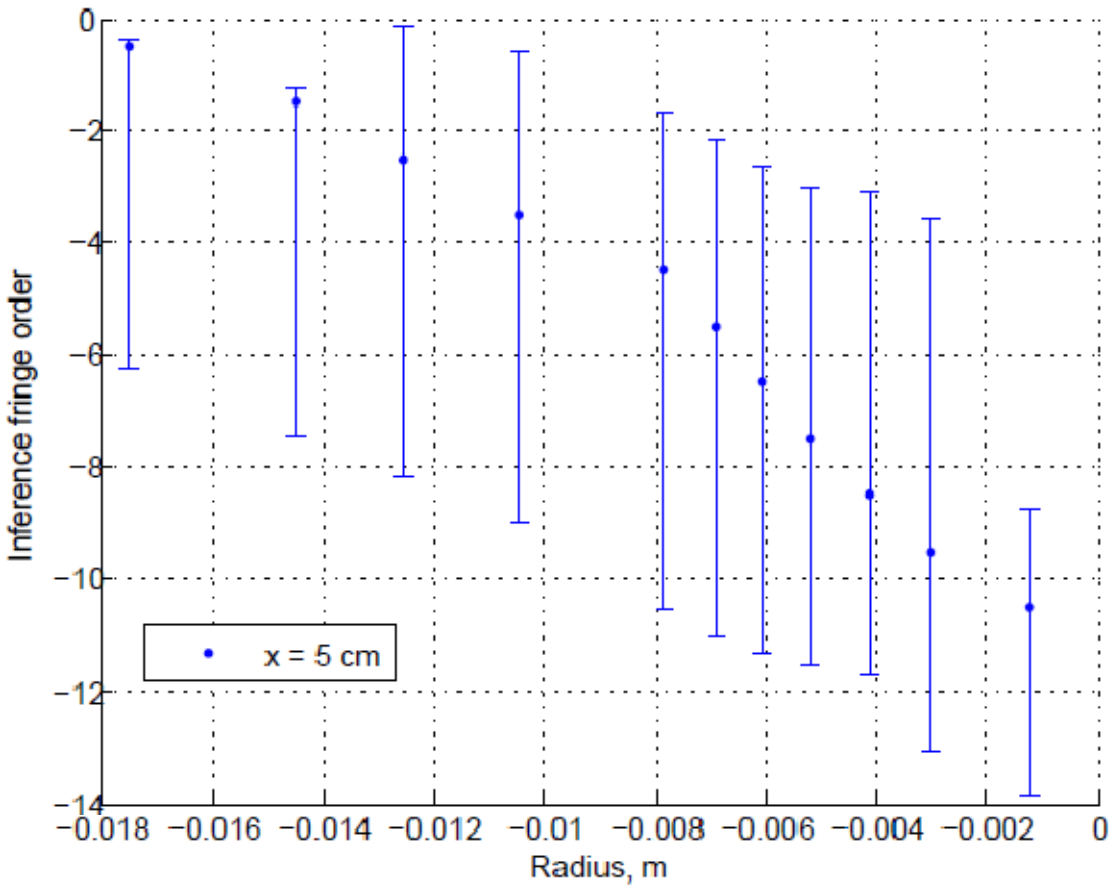
For the model uncertainty and measurement uncertainty to be added to the scenario uncertainty bars at the two measurement heights, the uncertainty bars in Figure 24 had to be transformed to vertical bars by interpolation as explained in the scenario uncertainty section. Adding the model uncertainty and measurement uncertainty to the transformed bars of Figure 24 resulted in Figure 28 with a maximum uncertainty of -7.96 fringes or 75.8% of total number of measured fringes. This range of uncertainty is very large, resulting mainly from large uncertainty due to bias.

Bias is large for this system due to the large ranges of experimental parameter uncertainties. The uncertainty in the experimental parameters (seen in Table 2) comes from historical weather data for temperature and pressure along with industrial standards for laser wavelength and the compositions of the species. Due to the firsthand accounts of exactly what time in the day these experiments were performed, a ten degree difference in temperature is used in the temperature parameter uncertainty. If the experiments were to be run again in the present day, the ranges on these parameters would be reduced significantly and a smaller area for the analytical solution would be seen for the bias uncertainty.



(a)

Figure 28: Transformed error bars for time-averaged fringe profiles. (a) x = 3 cm, (b) x = 5 cm



(b)

Figure 28: Continued.

CFD simulations of HI experiments

The CFD simulations were performed by Weston Eldredge of the Institute for Clean and Secure Energy at the University of Utah with Professor Jeremy Thornock serving as consultant. The CFD simulation domains were 13.8 cm cubed. The program ARCHES was used to produce the helium CFD simulations. ARCHES works by solving the conservation equations using the finite volume approach and a staggered grid. To achieve laminar flow in ARCHES, as was necessary in this study, parameters in the Smagorinsky subgrid scale model have to be set to zero. Additional code also had to be written by Mr. Elderedge and Professor Thornock to change the Schmidt number to the correct laminar value using the calculated diffusivity of helium in air. An image from the CFD simulation is shown in Figure 29. A total of 100^3 nodes was determined as sufficient resolution to solve the domain and was used to produce the CFD simulation results.

The uncertainty in the experimental parameters was run through the CFD simulation to produce bounds in the CFD simulation output. To achieve the semblance of an open-system in the closed-system CFD simulations, a co-axial flow of air was introduced outside the port radius.

The factors most affecting interference fringe orders were temperature, helium velocity, and the co-flow velocity of air. A Box-Behnkin design like that used for the sensitivity analysis was employed and the bounds are shown in Table 3. The co-flow velocity was based off of the nominal helium flow rate with the lower bound at 10% of The temperature range came from historical weather data [13].

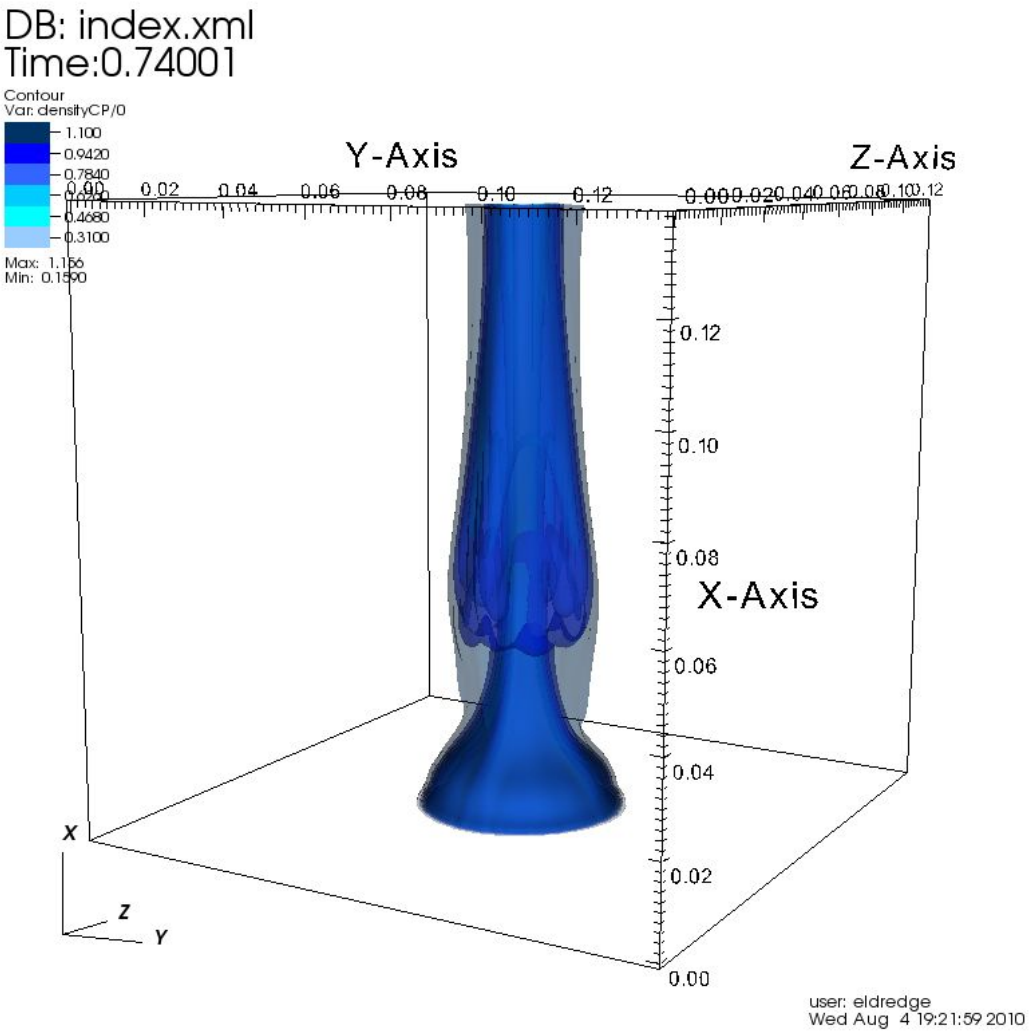


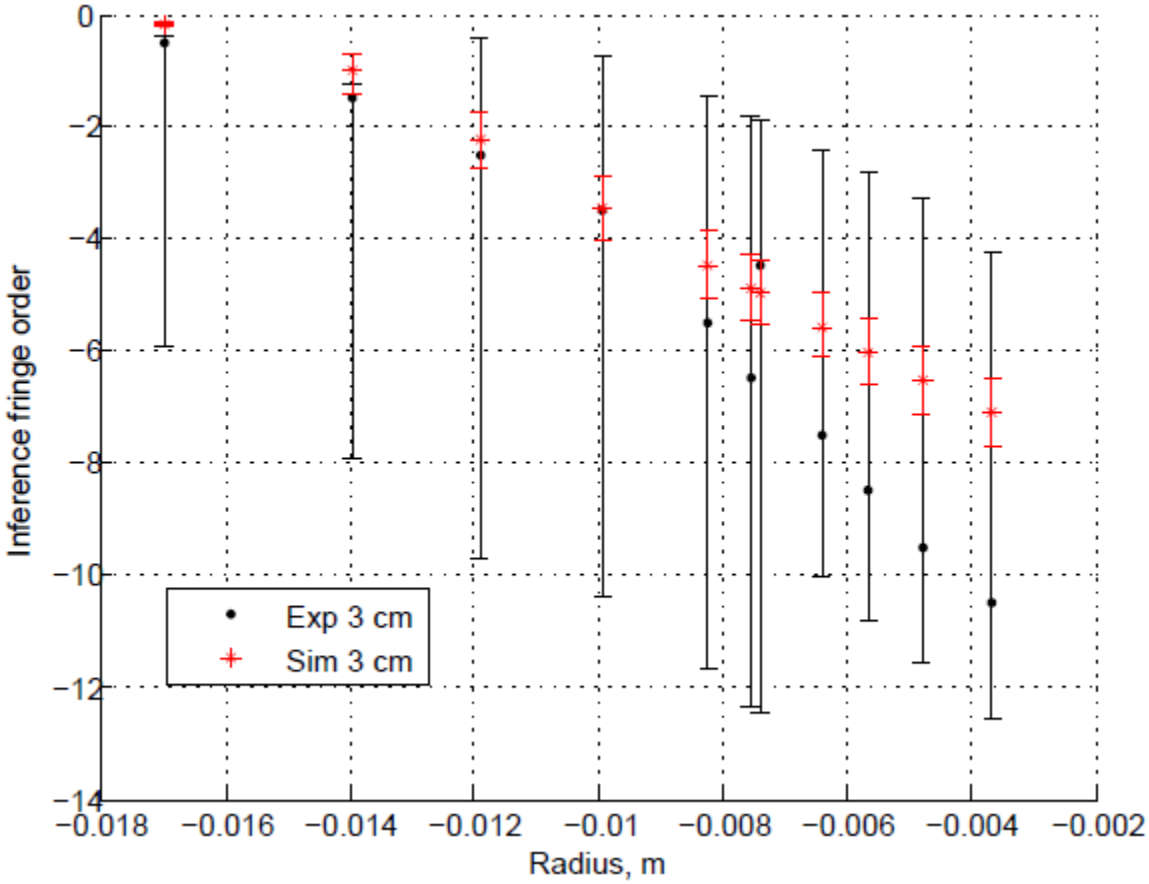
Figure 29: Image of the CFD simulation domain.

Table 3: Box-Behnkin for CFD simulations

Parameter	Lower bound	Middle	Upper bound
Temperature, K	295.15	305.15	315.15
Helium, m/sec	0.1215	0.135	0.1485
Air co-flow, m/sec	0.0135	0.0405	0.0675

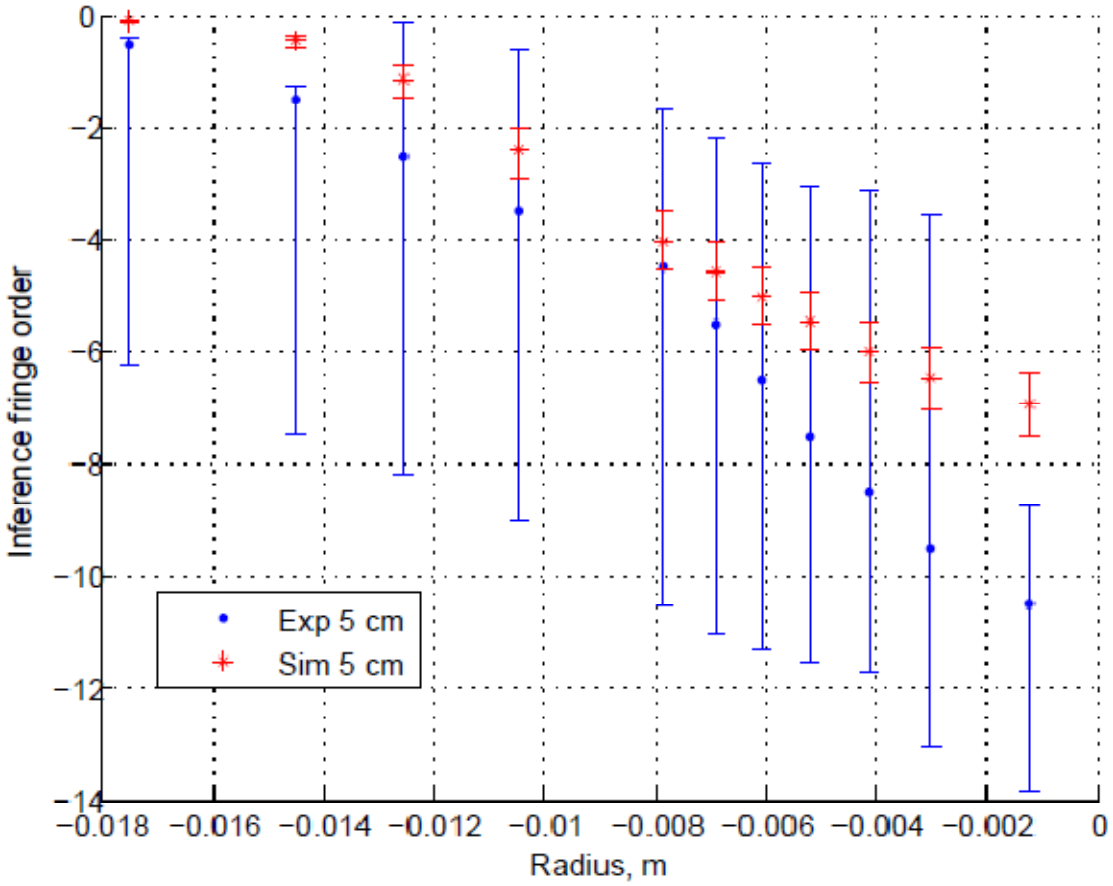
The helium nominal velocity was based on a flow rate of $2.25 \cdot 10^{-4} m^3/sec$ with the bounds as 10% of nominal flow rate in difference.

After all the CFD simulation cases were run, the data were transformed into the quantity of interest through the Gladstone-Dale equation (Equation 1) and the optical path length difference equation (Equation 3) to obtain simulated interferograms. The simulated interferograms were then time-averaged and the mean locations as well as the bounds were graphed in Figure 30 with the experimental results and uncertainty bounds. The results shown are only a preliminary result, which lacks the discretization error for the CFD simulations.



(a)

Figure 30: Experimental and simulated results with uncertainty bounds. (a) x = 3 cm, (b) x = 5 cm



(b)

Figure 30: Continued.

Limitations of This Study and Method

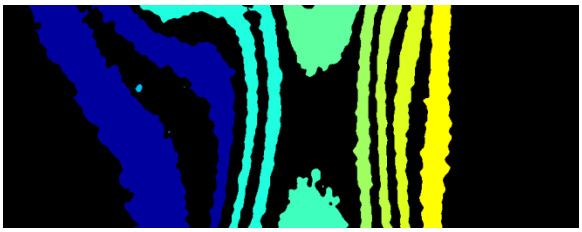
This study proposed a novel method for the validation of HI CFD simulations using optical measurements with a more quantitative focus than those previously employed. Although this analysis attempted to thoroughly identify all sources of uncertainty, some sources of uncertainty remained incalculable. Because the experiments were performed 25 years ago at a different institution, much of the insight into the uncertainty of experimental conditions and information from first-hand accounts have been lost.

The validation method proposed in this study has some limitations when applied to other systems. The quantitative comparison only works in systems where the uncertainty can be quantified. If the error and uncertainty cannot be quantified, then the qualitative comparison is the appropriate method for the comparison.

Additional Observations

An interesting phenomenon was observed during the processing of the interferograms. For each data set, a specific number of fringes were observed at the pinch point. The pinch point is observed between two puffs when the plume thins to the minimum number of fringes. The number of fringes at the pinch point ranges from four at the lower flow rates for Helium_01 ($2.25 \cdot 10^{-4}$ m^3/sec), Helium_02 ($2.0\backslash \cdot 10^{-4}$ m^3/sec), five for Helium_04 ($2.3 \cdot 10^{-4}$ m^3/sec), and six at the highest flow rate for Helium_03 ($2.5 \cdot 10^{-4}$ m^3/sec), as seen in Figure 31.

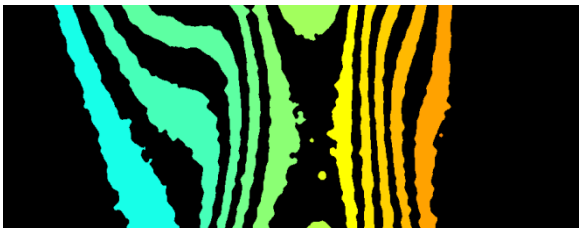
Since each fringe represents a change in density, it can be surmised that at each flow rate, there was a steady flow of some amount of helium from the port, making a



(a)

(b)

(c)



(d)

Figure 31: Pinch point fringe observations: (a) $2.25 \cdot 10^{-4}$ m^3/sec, (b) $2.0 \cdot 10^{-4}$ m^3/sec, (c) $2.5 \cdot 10^{-4}$ m^3/sec, (d) $2.3 \cdot 10^{-4}$ m^3/sec

sheath through which the puff flowed. It was interesting to note that Helium_01 and Helium_02 have the same number of exterior fringes, though their flow rates are different. It was surmised that two data sets that have close flow rates would have the same flow structures because Helium_01 and Helium_04, which have the closest flow rates, have similar flow structures. It was first theorized that the flow rates of the first two data sets might actually have been closer; however, upon visually comparing interferograms from each data set, it was dismissed as they have very different flow structures. It was then theorized that the number of exterior fringes had thresholds between each flow rate. Helium_02 was surmised to be well in the region of the four external fringe limit with Helium_01 pushing the upper boundary. Further study into this phenomenon could result in another measurement for validation where the flow structures could provide quantities of interest such as aspect factors.

Future Work

As discussed previously, the consistency analysis was not performed as part of this thesis due to time constraints. The consistency analysis will be performed by Weston Eldredge. The data collaboration software to perform the consistency analysis will be provided in the future by Feeley et al. from the University of California at Berkeley. The consistency analysis will then take the uncertainty quantified in the Data set from this thesis and use Lagrangian multipliers to analyze the sensitivity of the consistency to the quantity of interest's uncertainty [11]. The consistency analysis will output a range of uncertainty that gives consistent results between the experiments and CFD simulations, along with a quantified value of consistency.

To improve upon the methods laid out in this study, the addition of another simultaneous measurement at an angle to the original (tomographic holographic interferometry) would increase the resolution and reduce image-processing uncertainty. Also, performing these experiments in the modern day with a digital camera would eliminate the need for registration, greatly reducing the image-processing time, and give insight into the experimental parameters. The temperature, pressure, boundary wind, and other experimental conditions could be quantified with direct measurements around the system rather than relying on the closest weather station. Were the experiments to be run again, they could be geared to work better with the image-processing. An analysis of the camera speed could be made in which to reduce the grainy noise observed in the interferograms by decreasing the speed. Image-processing time would be decreased with the improvement of the resolution in the images at slower camera speeds.

The scope of the comparison could be broadened to validate flow structure by changing the data sets. If a data set was defined as the same image from each puff, the shape and position of the toroidal vortex could be tracked and quantified. The shape could be verified through the aspect ratio, and the vortice positions could be verified by fringe location.

APPENDIX A

IMAGE-PROCESSING GUIDE

LOCATION OF CODES AND DIRECTORY FOR IMAGES:

The codes necessary for image-processing can be seen in Appendix B. Before running these codes a directory must be chosen such that MATLAB can find the images to process.

INTENSITY EQUALIZATION

Purpose: Set an brightness, light intensity for the entire data set to improve registration.

Code: brighten_every_image.m

Instructions:

Load the first image of the data set into the directory

Load the other images into directory that you want to process

Run code

Input starting image number when prompted

Check that it counts up each image properly on line 65 (i.e. num + 1 for every image processing and num + 5 for every fifth image processing

Copy output images from output directory (default set as MATLAB in my documents folder) into folder on hard drive.

REGISTRATION

Purpose: Register the images to a common axis.

Code: register_every_image.m

Instructions:

Load the first image of the data set into the directory

Load the other images into directory that you want to process

Run code

Input starting image number when prompted

Check that it counts up each image properly (i.e. num + 1 for every image processing and num + 5 for every fifth image processing

Copy output images from output directory (default set as MATLAB in my documents folder) into folder on hard drive.

CROPPING AND FILTERING

Purpose: Align images to a common axis. This code is only necessary for data sets where the images “bounce” or move from the common axis determined visually by watching a movie or rapid sequence of images.

Code: INF_IPloop_Helium_0*_every_image.m

Instructions:

Make sure the m-file corresponds to the data set being processed

Different data sets have different values in functions to properly filter their images

Load the first image of the data set into the directory

Load the other images into directory that you want to process

Run code

Input starting image number when prompted

Check that it counts up each image properly on line 70 (i.e. num + 1 for every image processing and num + 5 for every fifth image processing

Copy output images from output directory (default set as MATLAB in my documents folder) into folder on hard drive

RESOLVE FRINGES

Purpose: Hand filter images so that they can be read by the next code. The filtering employed by MATLAB is not sophisticated enough to clear out all the noise and totally resolve all the fringes.

Load each image individually into a photoshop program

Add and delete pixels to create continuous fringes

Clear all noise on the side of the images

Save images in a folder on the hard drive

EXTRACT ONE PIXEL SECTIONS AT 3 AND 5 CM FOR STATISTICAL ANALYSIS

Purpose: Extract data from the 3 and 5 cm heights into a excel file for post-processing.

Code: New_Dataset_PDF_Maker.m

Instructions:

Load all the images of a statistical data set into the directory before running this code

Set which height to measure at in line 32 by specifying ymin = 0 for 3cm and ymin = 357 for 5cm

Run code

Save Excel files to a folder on the hard drive

Save matrices in Excel files to MATLAB files

Clear out the workspace (command: clear all)

Create a new variable or matrix (button under “Workspace”)

Label the new variable histograms as “hist” and the PDFs as “PDF_avg”

Copy the matrix from the Excel file

Copy in Helium_01_LHS

Copy in Helium_01_RHS

Copy in Helium_04_LHS

Copy in Helium_04_RHS

Save the new variable in a folder

Save the histogram and PDFs corresponding to their heights

There should be one histogram file labeled, “hist_3cm” and PDF file labeled,

“PDF_3cm”

Same for 5cm

A total of four files

STATISTICAL CODES

Purpose: Garner statistical data on the data at 3 and 5 cm heights

Code: new_bin_Overall_avg_PDF.m

Instructions:

Drag the PDF_avg matrix at the height being analyzed into Matlab’s Workspace

Run the code to get the Overall average PDF for the entire data population for each fringe

Save images of PDFs

The mean locations are shown in the variable, avg_avg_sum

The deviation in locations for each fringe are shown in variable, dev_sum_avg

APPENDIX B

IMAGE-PROCESSING CODES

registration_every_image.m

```
%Kerry Kelly, Base Code, Laurie Marcotte, Modifications
%_____________________________________________________
% Image Registration for Helium set 01-04 due to their need of sharpening.
% This code reads all images in folder Images3, puts them into a matrix and
% sharpens and registers each image. This reduces the bouncing and
% shuffling seen in movies of the images that would greatly reduce the
% accuracy of the final measurements.
%_____________________________________________________
%Registration Code
addpath('C:\Work Files\images 8-21-09\Images3')
% Line 13: Adds the folder to those able to be accessed by MatLab.
fileFolder = fullfile('C:','Work Files','images 8-21-09', 'Images3');
dirOutput = dir(fullfile(fileFolder,'Helium_notbright_01_*.tif'));
fileNames = {dirOutput.name}';
numFrames = numel(fileNames);
% Lines 16-19: show the code exactly where the files to be registered
```

```
% are. Line 19 will later dictate the number of repetitions through
% the for loops that will be made.
I = imread(fileNames{1});
sequence = zeros([size(I) numFrames],class(I));
Reg_sequence = zeros([size(I) numFrames],class(I));
% Lines 24-26: Image 1 is read to set up suitable arrays that are
% clear of any other images or numbers in which the images can be
% placed and registered.
for p = 1:numFrames
I4 = imread(fileNames{p});
% figure,subplot(2,1,1);c
% imshow(I4); title('Original Image');
H = fspecial('unsharp');
sharpened = imfilter(I4,H,'replicate');
% subplot(2,1,2);
% imshow(sharpened); title('Sharpened Image');
sequence(:,:,p)=sharpened;
end
% Lines 32-49: This is a "for loop" that takes each image, displaces
% it by a few pixels, overlays the displaced image on the orginal,
% makes a filter from the blurred image and filters the original
% image to sharpen it. This code comes straight from the example for
% "fspecial" in the MatLab Help.
```

```
num = input('Enter starting image number (not including base image):');
% Line 56: the value "num" is used to write the registered images as
% their correct filenames instead of letting MatLab number them 1,2,3
% etc. without regard for the fact that they are images 100 or 565 etc.
for p = 2:numFrames
inf1=sequence(:,:,1);
Reg_sequence(:,:,1)=inf1;
inf2=sequence(:,:,p);
[input_points, base_points]=cpselect(inf2,inf1,'wait',true);
input_points
base_points
input_points_corr=cpcorr(input_points,base_points,inf2,inf1);
mytform=cp2tform(input_points,base_points, 'linear conformal');
Reg_Image=imtransform(inf2,mytform, 'XData', [1 size(inf1,2)],'YData', [1
size(inf1,1)]);
Reg_sequence(:,:,p)=Reg_Image;
imwrite(Reg_Image, sprintf('Heliumreg_01_00%04d.tif',num));
num=num+1
end
% Lines 61-74: the "for loop" that registers the images. Taking the
% first image in the set, always Helium_0*_001, it allows the user to
% select points on the image using the function "cpselect"(Usually
% the pan's edges and a fixed dot) on both images to align them in
% the same set of axises using function "cpcorr" to reduce bouncing
```

```
% and shuffling.
%_____________________________________________
close all
clear all
clc
```

Inf_IPloop_Helium_01_every_image.m

```
%Kerry Kelly, Base Code, Laurie Marcotte, Modifications
% _____

% This code crops and filters the image before it is pulled into Photoshop

% to be hand filtered and corrected. Subsequent codes that have the same

% name are tailored to fit each set of Helium 01-04. This tailoring is

% comprised of assessing the values of im2bw ranging from 0-1 and bwmorph

% which has a variety of filters.

% _____

%Crop Code

addpath('C:\Work Files\images\Images3')

% Line 9: Adds the folder to those able to be accessed by MatLab.

fileFolder = fullfile('C:','Work Files','images', 'Images3');

dirOutput = dir(fullfile(fileFolder,'Heliumreg_01_*.tif'));

fileNames = {dirOutput.name}';

numFrames = numel(fileNames);

% Lines 12-15: show the code exactly where the files to be registered

% lay. Line 15 will later dictate the number of repetitions through

% the for loops that will be made.

I = imread(fileNames{1});

IP_sequence = zeros([size(I) numFrames],class(I));

% Lines 25-26: The first image is read to determine the class and

% size of all the images in the File Folder. These values are then

% used to clear a matrix and set it to the size and class of the
```

```
% first image in the set. This set is usually called preallocating
% the array.
for p = 1:numFrames
I = imread(fileNames{p});
IP_sequence(:,:,p) = I;
end
% Lines 33-36: Using the array(matrix) preallocated in line 26 this
% "for" loop inputs each image in the file folder into the array.
num = input('Enter starting image number:');
% Line 40: Matlab names files from 1 on without regard to any numbers
% in their titles thus when we rename the files later we need to
% manually number them using and input in line 40 and a counter to
% order the files correctly.
rect=[558 152 926 356];
% Line 46: This rectangle defines the part of the image that will be
% cropped. This sets the crop 3 cm from the top of the pan with a
% height of 2 cm. The format of this value is: rect = [xmin ymin
% width height]
for p = 1:numFrames
I2=adapthisteq(IP_sequence(:,:,p),'NumTiles',[8,8],'ClipLimit',0.015);
BW=im2bw(I2,0.82);
BW2=bwmorph(BW,'clean',10);
BW3=bwmorph(BW2,'fill',10);
```

```
BW4=bwmorph(BW3,'majority');

BW5=bwmorph(BW4,'thin');

BW6=medfilt2(BW5,[8 8]);

BW7=imcomplement(BW6);

G2=imcrop(BW7,rect);

BWN=bwlabeln(G2);

RGB=label2rgb(BWN, @jet, 'k');

imwrite(RGB, sprintf('Helium_01_crop_%04d.tif',num));

num=num+1

end

% Lines 52-66: This is the "for" loop that filtered, cropped, ran

% connected components,and re-saved the images in the matrix

% IP_sequence. The filter clean gets rid of one pixel spots, fill

% gets rid of spots that are in the middle of the spaces in between

% the fringes. Majority cleans the edges of the fringes by getting

% rid of pixels unlike the majority around them. The filter thin

% makes the spaces between the fringes smaller. Since the fringes

% are black and the spaces white the code, using the function

% imcomplement, turns the fringes white and the spaces between black.

% The bwlabeln function uses connected components to distinguish
% between the fringes.
% close all
% clear all
% clc
```

Inf_IPloop_Helium_02.m

```
addpath('F:\BackupC\CSAFE\images\Images3')
% Create an array of interferometry filenames
fileFolder = fullfile('F:','BackupC','CSAFE','images', 'Images3');
dirOutput = dir(fullfile(fileFolder,'Heliumreg_02_*.tif'));
fileNames = {dirOutput.name}';
numFrames = numel(fileNames);
%load Reg_sequence
I = imread(fileNames{1});
% Preallocate the array
IP_sequence = zeros([size(I) numFrames],class(I));
%IP_sequence(:,:,1) = I;
% Create image sequence array
for p = 1:numFrames
I = imread(fileNames{p});
IP_sequence(:,:,p) = I;
end
%Enter starting image number, not including image 1 the base image
num = input('Enter starting image number:');
%Note - you need to change the crop size for different image sequences
%rect = [xmin ymin width height]
rect=[558 152 926 356];
%Perform image processing and cropping
```

```
for p = 1:numFrames

I2=adapthisteq(IP_sequence(:,:,p),'NumTiles',[9 9],'ClipLimit',0.015);

% figure, imshow(I2); title('Original');

BW=im2bw(I2,0.50); %Note - you may want to change the threshold value for other
sequences

% figure, imshow(BW); title('im2bw');

BW2=medfilt2(BW,[9 9]);

% figure, imshow(BW2); title('Filtered');

BW3=bwmorph(BW2,'clean',10);

% figure, imshow(BW3); title('cleaned');

BW4=bwmorph(BW3,'fill',10);

% figure, imshow(BW4); title('filled');

BW5=bwmorph(BW4,'bridge',3);

% figure, imshow(BW5); title('bridge');

BW6=bwmorph(BW5,'majority');

% figure, imshow(BW5); title('Omajoritized');

BW7=bwmorph(BW6,'thin');

% figure, imshow(BW6); title('Thinned');

%Use complement of image, note black lines are the interesting ones

BW8=imcomplement(BW7);

% figure, imshow(BW7); title('Color Inverted');

%Crop image

G2=imcrop(BW8,rect);

% figure, imshow(G2); title('Cropped');
```

```
%Connected components

x=[-2.5 2.5];

y=[-1.54 -1.14];

BWN=bwlabeln(G2);

RGB=label2rgb(BWN, @jet, 'k');

% figure, imshow(RGB); title('Colored');

%I5=imcrop(Reg_sequence(:,:,1),rect);

% figure, image(RGB, 'XData', x, 'YData', y), axis image, colormap(jet(25)),
title(sprintf('Image to save %d',p))

%Select y location that you wish to write to a data file

%BWN2=imcrop(BWN, [1 1 278 0]);

%figure, imshow(BWN2)

%dlmwrite('He_data3', BWN2);

%imwrite(RGB, fileNames2{p});

%BWN3=imcrop(BWN, [1 21 278 0]);

%figure, imshow(BWN3)

%dlmwrite('He_data_d25', BWN3, '-append');

imwrite(RGB, sprintf('Helium_02_crop_00%d.tif',num));

num=num+5;

end
```

Inf_IPloop_Helium_03.m

```
addpath('F:\BackupC\CSAFE\images\Images3')
% Create an array of interferometry filenames
fileFolder = fullfile('F:','BackupC','CSAFE','images', 'Images3');
dirOutput = dir(fullfile(fileFolder,'Heliumreg_03_*.tif'));
fileNames = {dirOutput.name}';
numFrames = numel(fileNames);
%load Reg_sequence
I = imread(fileNames{1});
% Preallocate the array
IP_sequence = zeros([size(I) numFrames],class(I));
%IP_sequence(:,:,1) = I;
% Create image sequence array
for p = 1:numFrames
I = imread(fileNames{p});
IP_sequence(:,:,p) = I;
end
%Enter starting image number, not including image 1 the base image
num = input('Enter starting image number:');
%Note - you need to change the crop size for different image sequences
% rect=[715 718.5 500 111]%set crop size
%rect = [xmin ymin width height]
rect=[558 152 926 356];
```

```
%Perform image processing and cropping

for p = 1:numFrames

I2=adapthisteq(IP_sequence(:,:,p),'NumTiles',[9 9],'ClipLimit',0.015);

% figure, imshow(I2); title('Original');

BW=im2bw(I2,0.9225); %Note - you may want to change the threshold value for other
sequences

% figure, imshow(BW); title('im2bw');

BW2=bwmorph(BW,'clean',10);

% figure, imshow(BW2); title('cleaned');

BW3=bwmorph(BW2,'fill',10);

% figure, imshow(BW3); title('filled');

BW4=bwmorph(BW3,'bridge');

% figure, imshow(BW4); title('bridge');

% BW5=bwmorph(BW4,'majority');

% figure, imshow(BW5); title('Omajoritized');

BW6=bwmorph(BW4,'thin');

% figure, imshow(BW6); title('Thinned');

BW7=medfilt2(BW6,[9 9]);

% figure, imshow(BW7); title('Filtered');

%Use complement of image, note black lines are the interesting ones

BW8=imcomplement(BW7);

% figure, imshow(BW8); title('Color Inverted');

%Crop image

G2=imcrop(BW8,rect);
```

```
% figure, imshow(G2); title('Cropped');

%Connected components

x=[-2.5 2.5];

y=[-1.54 -1.14];

BWN=bwlabeln(G2);

RGB=label2rgb(BWN, @jet, 'k');

% figure, imshow(RGB); title('Colored');

%I5=imcrop(Reg_sequence(:,:,1),rect);

% figure, image(RGB, 'XData', x, 'YData', y), axis image, colormap(jet(25)),
title(sprintf('Image to save %d',p))

%Select y location that you wish to write to a data file

%BWN2=imcrop(BWN, [1 1 278 0]);

%figure, imshow(BWN2)

%dlmwrite('He_data3', BWN2);

%imwrite(RGB, fileNames2{p});

%BWN3=imcrop(BWN, [1 21 278 0]);

%figure, imshow(BWN3)

%dlmwrite('He_data_d25', BWN3, '-append');

imwrite(RGB, sprintf('Helium_03_crop_00%d.tif',num));

num=num+5

end

close all
clear all
clc
```

Inf_IPloop_Helium_04.m

```
addpath('C:\Work Files\images\Images3')
% Create an array of interferometry filenames
fileFolder = fullfile('C:','Work Files','images', 'Images3');
dirOutput = dir(fullfile(fileFolder,'Heliumreg_04_*.tif'));
fileNames = {dirOutput.name}';
numFrames = numel(fileNames);
%load Reg_sequence
I = imread(fileNames{1});
% Preallocate the array
IP_sequence = zeros([size(I) numFrames],class(I));
%IP_sequence(:,:,1) = I;
% Create image sequence array
for p = 1:numFrames
I = imread(fileNames{p});
IP_sequence(:,:,p) = I;
end
%Enter starting image number
num = input('Enter starting image number:');
%Note - you need to change the crop size for different image sequences
% rect=[715 718.5 500 111]%set crop size
%rect = [xmin ymin width height]
rect=[558 152 926 356];
```

```
%Perform image processing and cropping

for p = 1:numFrames

I2=adapthisteq(IP_sequence(:,:,p),'NumTiles',[9 9],'ClipLimit',0.015);

BW=im2bw(I2,0.50); %Note - you may want to change the threshold value for other
sequences

BW2=bwmorph(BW,'clean',10);

BW3=bwmorph(BW2,'fill',10);

BW4=bwmorph(BW3,'majority');

BW5=bwmorph(BW4,'thin');

%Use complement of image, note black lines are the interesting ones

BW6=medfilt2(BW5,[9 9]);

BW7=imcomplement(BW6);

%Crop image

G2=imcrop(BW7,rect);

%Connected components

x=[-2.5 2.5];

y=[-1.54 -1.14];

BWN=bwlabeln(G2);

RGB=label2rgb(BWN, @jet, 'k');

%I5=imcrop(Reg_sequence(:,:,1),rect);

% figure, image(RGB, 'XData', x, 'YData', y), axis image, colormap(jet(25)),
title(sprintf('Image to save %d',p))

%Select y location that you wish to write to a data file

%BWN2=imcrop(BWN, [1 1 278 0]);
```

```
%figure, imshow(BWN2)

%dlmwrite('He_data3', BWN2);

%imwrite(RGB, fileNames2{p});

%BWN3=imcrop(BWN, [1 21 278 0]);

%figure, imshow(BWN3)

%dlmwrite('He_data_d25', BWN3, '-append');

imwrite(RGB, sprintf('Helium_04_crop_00%d.tif',num));

num=num+5

end

close all

clear all

clc
```

New_Dataset_PDF_Maker.m

```
% This code makes a average PDF and histogram for each data set and prints
% them as excel for backups. These excel files should then be compiled
% into a larger matrix composed of all the sets PDFs and histograms.
%Load SA data into workspace before running for the specific data set
addpath('C:\Work Files\images\Images3')
% Create an array of interferometry filenames
fileFolder = fullfile('C:','Work Files','images', 'Images3');
dirOutput = dir(fullfile(fileFolder,'Helium_04_*.tif'));
fileNames = {dirOutput.name}';
numFrames = numel(fileNames);
%Check first Image
I = imread(fileNames{1});
% Preallocate the array
IP_sequence = zeros([size(I),numFrames],class(I));
% Create image sequence array
for p = 1:numFrames
I2 = imread(fileNames{p});
IP_sequence(:,:,:,p) = I2;
end
%Define sensitivity analysis error parameters for the data set
SA = SA_5cm(:,:,0);
% For no SA error conf = 0, hconf use conf = 1, for lconf use conf = 2
```

```
conf = 2;
%Define piece to be cut
rect=[0 0 926 1];%rect = [xmin ymin width height]
% 5 cm - ymin=0, 3 cm - ymin=357
I3=imcrop(IP_sequence(:,:,:,1),rect);
BW = im2bw(I3, 0.05);
BWN2=bwlabeln(BW);
[r,c1,colors]=size(BWN2);
s = regionprops(BWN2, 'Centroid');
centroids=cat(1, s.Centroid);
newc = c1*2;
doubled_image = zeros([r,newc],class(BWN2));
all_doubled_image = zeros([size(doubled_image) numFrames],class(BWN2));
BW1 = zeros(size(BW));
BWN3= zeros(size(BWN2), class(BWN2));
centroids1= zeros(25,2);
%Preallocate more arrays
IP_sequence2 = zeros([size(I3),numFrames],class(I3));
IP_sequence3 = zeros([size(I3),numFrames],class(I3));
BWN = zeros([size(I3),numFrames],class(I3));
BWN_LHS = zeros([size(doubled_image),numFrames],class(BWN2));
BWN_RHS = zeros([size(doubled_image),numFrames],class(BWN2));
BWNLHS = zeros(size(doubled_image),class(BWN2));
```

```
BWNRHS = zeros(size(doubled_image),class(BWN2));

Radius = zeros(size(doubled_image));

x = zeros(numFrames,1);

for w = 1:numFrames

%Skeletonization

centroids1= zeros(25,2);

doubled_image = zeros([r newc],class(BWN2));

I4=imcrop(IP_sequence(:,:,:,w),rect);

IP_sequence3(:,:,:,w) = I4;

BW1 = im2bw(IP_sequence3(:,:,:,w), 0.05);

BWN3=bwlabeln(BW1);

[r,c3,colors]=size(BWN3);

s = regionprops(BWN3, 'Centroid');

centroids1=cat(1, s.Centroid);

%Assign fringes to radial bins including SA error when applicable

for v = 1:max(BWN3);

if conf > 0

high = max(BWN3);

half= high/2;

interger=floor(half);

part=half-interger;

if part == 0;

x(w,1) = 0; %Even
```

```
outer_num=high/2;
else
x(w,1) = 1; %Odd
outer_num=interger+1;
end
if v<=outer_num
n = centroids1(v,1)*2+ SA(conf,v);
elseif v>outer_num
fringe = high-v+1;
n = centroids1(v,1)*2+ SA(conf,fringe);
end
doubled_image(1,n)=v;
elseif conf==0;
n = centroids1(v,1)*2;
doubled_image(1,n)=v;
end
end
all_doubled_image(:,:,w)= doubled_image;
%Negative Half numbering
BW1 = zeros(size(BW));
BWN3= zeros(size(BWN2), class(BWN2));
BW = im2bw(all_doubled_image(:,:,w), 0.05);
BWN2=bwlabeln(BW);
```

```
[m,p] = size(BWN2);

[m,p] = size(BWN2);

high = max(BWN2);

half= high/2;

interger=floor(half);

part=half-interger;

if part == 0;

x(w,1) = 0; %Even

outer_num=high/2;

else

x(w,1) = 1; %Odd

outer_num=interger+1;

end

for q=1:p

if BWN2(:,q)==0

BWNLHS(:,q)=0;

elseif BWN2(:,q)>0 && BWN2(:,q)<outer_num + 1

n=BWN2(:,q);

m=n-1;

BWNLHS(:,q)=-0.5 - m;

end

end

n=0;
```

```
m=0;
for q=1:p
if BWN2(:,q)==0
BWNRHS(:,q)=0;
elseif part == 0 && BWN2(:,q)>outer_num && BWN2(:,q)< high + 1
n=BWN2(:,q);
m=n-outer_num;
Fringe_num=n-2*m;
BWNRHS(:,q)=-0.5-Fringe_num;
elseif part~= 0 && BWN2(:,q)>outer_num && BWN2(:,q)< high + 1
n=BWN2(:,q);
m=n-outer_num;
Fringe_num=n-2*m;
BWNRHS(:,q)=0.5-Fringe_num;
end
end
BWN_LHS(:,:,w)=BWNLHS;
BWN_RHS(:,:,w)=BWNRHS;
BWNLHS = zeros(size(doubled_image),class(BWN2));
BWNRHS = zeros(size(doubled_image),class(BWN2));
end
%++++++++++++++++++++++++++++++++++++++
% Statistics
```

```
high = 11;

%Setup up information for statistics compilation

[r c2] = size(doubled_image);

pixelspercm = 178;

radius = -483/pixelspercm; %cm

cnew = radius*-1*pixelspercm*4+1;

%Multiplied by 4 to account for doubled pixels and asymetricity

j=0;

%Make the matrixes flipable on the pan axis

csym = c2+81; %Inc. columns to make it symmetric on pan axis

for c = c2:csym

BWN_LHS(:,c,:)=0;%Add columns of 0's to matrices

BWN_RHS(:,c,:)=0;%When folded each half of the matrix is 966 pixels in width

end

%Flip the RHS matrix

for w = 1:numFrames;

BWN_RHS(:,:,w) = fliplr(BWN_RHS(:,:,w));

end

%Add in assymetric center fringe for odd numbered images

for w = 1:numFrames;

if x(w)==1; %if the image has an odd number of fringes

inner = min(BWN_LHS(:,:,w));

[row column] = find(BWN_LHS(:,:,w)==inner); %find the inner fringe in the LHS
matrix
```

```
BWN_RHS(row,column,w)=BWN_LHS(row,column,w);
%Assign inner fringe as same for LHS and RHS
end
end
%Write BWN_* matrices to excel backups
for w = 1:numFrames
dlmwrite('Helium_skel_04_5cm_LHS_5-10_lconf.xls', BWN_LHS(:,:,w),'- append');
dlmwrite('Helium_skel_04_5cm_RHS_5-10_lconf.xls', BWN_RHS(:,:,w),'- append');
end
%Compile radius matrix
for i=1:cnew
pixelspercm = 178; %pixels per centimeter
rradius = radius +j*0.5/pixelspercm;
Radius(1,i)=rradius;
j=j+1;
end
n=0;
im=0;
totalnumber_LHS = zeros(high,1);
total_number_LHS = zeros(high,csym);
totalnumber_RHS = zeros(high,1);
total_number_RHS = zeros(high,csym);
% Total number calculation - LHS
```

```
for im=1:w;
for n=1:csym;
fringe0=-0.5;
for valuef=1:high
if BWN_LHS(:,n,im)== fringe0;
totalnumber_LHS(valuef)=totalnumber_LHS(valuef)+1;
value = total_number_LHS(valuef,n);
total_number_LHS(valuef,n) = value+1;
end
fringe0=fringe0-1;
end
end
end
% Total number calculation - RHS
n=0;
im=0;
for im=1:w;
for n=1:csym;
fringe0=-0.5;
for valuef=1:high
if BWN_RHS(:,n,im)== fringe0;
totalnumber_RHS(valuef)=totalnumber_RHS(valuef)+1;
value = total_number_RHS(valuef,n);
```

```
total_number_RHS(valuef,n) = value+1;
end
fringe0=fringe0-1;
end
end
end
%Data for histogram
PDF_data_LHS = zeros(high,csym);
PDF_data_RHS = zeros(high,csym);
n=0;
v=0;
for v=1:high;
for n=1:csym;
if n<=csym;
PDF_data_LHS(v,n) = total_number_LHS(v,n);
PDF_data_RHS(v,n) = total_number_RHS(v,n);
else n>csym;
PDF_data_LHS(v,n) = 0;
PDF_data_RHS(v,n) = 0;
end
end
end
dlmwrite('Helium_hist_04_LHS_5cm_5-10_lconf.xls', PDF_data_LHS);
```

```
dlmwrite('Helium_hist_04_RHS_5cm_5-10_lconf.xls', PDF_data_RHS);
% Make the data for the average PDF - LHS
PDF_LHS = zeros(high,csym);
PDF_RHS = zeros(high,csym);
v=0;
n=0;
for n=1:csym;
for v=1:high;
PDF_LHS(v,n)=total_number_LHS(v,n)/totalnumber_LHS(v);
end
end
% Make the data for the average PDF - RHS
v=0;
n=0;
for n=1:csym;
for v=1:high;
PDF_RHS(v,n)=(total_number_RHS(v,n)/totalnumber_RHS(v));
end
end
avgPDF_data_LHS = zeros(high,csym);
avgPDF_data_RHS = zeros(high,csym);
n=0;
v=0;
```

```
for v=1:high;

for n=1:csym;

if n<=csym;

avgPDF_data_LHS(v,n) = PDF_LHS(v,n);

avgPDF_data_RHS(v,n) = PDF_RHS(v,n);

else n>csym;

avgPDF_data_LHS(v,n) = 0;

avgPDF_data_RHS(v,n) = 0;

end

end

end

dlmwrite('Helium_PDF_04_LHS_5cm_5-10_lconf.xls', avgPDF_data_LHS);

dlmwrite('Helium_PDF_04_RHS_5cm_5-10_lconf.xls', avgPDF_data_RHS);
```

new_bin_Overall_avg_PDF.m

```
%Average PDF for the entire population of data.
%Drag in PDF_avg matrix before starting code
% PDF_avg = all_SA_PDF_3cm;
[r c set] = size(PDF_avg);
n = 45; % number of pixels in a bin
length = round(c/n); % New length with new bin width
%Re-bin PDF_avg
nPDF_avg = zeros(r,length,set);
for sets = 1:set;
for v = 1:r;
bin = 1;
lbin = 1;
hbin = lbin + n;
for x = 1:c;
if bin>length
nPDF_avg(v,length,sets)= nPDF_avg(v,length,sets) + PDF_avg(v,x,sets);
end
if x>=lbin && x<hbin && bin<length
nPDF_avg(v,bin,sets) = nPDF_avg(v,bin,sets) + PDF_avg(v,x,sets);
end
if x==hbin && bin<length
lbin = lbin + n;
```

```
hbin = hbin + n;

bin = bin + 1;

nPDF_avg(v,bin,sets) = nPDF_avg(v,bin,sets) + PDF_avg(v,x,sets);

end

end

end

end

%Reassign row and column values for re-bin

[r c set] = size(nPDF_avg);

% Create S value matrix

start = -0.5;

svalues = zeros(r,1);

svalues(1,1) = start;

for q = 2:r;

svalues(q,1) = start-q+1;

end

radius = -483/178; %cm

j = 0;

%Compile radius matrix

Radius = zeros(1,length);

for i=1:length

pixelspercm = 178; %pixels per centimeter

rradius = radius +j*0.5*n/pixelspercm;
```

```
Radius(1,i)=rradius;

j=j+1;

end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Average PDF

% Summation

sum = zeros(size(nPDF_avg(:,:,1)));

sum_PDF_avg = zeros(size(sum));

for w=1:set;

sum = nPDF_avg(:,:,w);

sum_PDF_avg = sum_PDF_avg + sum;

end

% Total number of fringe occurances

fringe_sum_avg = zeros(r,1);

for q = 1:r;

for a = 1:c;

fringesum = sum_PDF_avg(q,a);

fringe_sum_avg(q,1) = fringe_sum_avg(q,1) + fringesum;

end

end

% Average PDF

tPDF_avg = zeros(size(sum_PDF_avg));

for q = 1:r;
```

```
for a = 1:c;
tPDF_avg(q,a) = sum_PDF_avg(q,a)/fringe_sum_avg(q,1);
end
end
for q = 1:r; % for each fringe
figure(q)
fringe = 0.5-q;
bar(Radius,tPDF_avg(q,:));
title (['Average PDF for interference fringe ',num2str(fringe)])
xlabel('Radial positions, cm'), ylabel('Probability')
end
%%Determine Average fringe location
avg_avg_sum = zeros(size(fringe_sum_avg));
for q = 1:r;
for a = 1:c;
meansum_avg = Radius(1,a)*tPDF_avg(q,a);
avg_avg_sum(q,1) = meansum_avg + avg_avg_sum(q,1);
end
end
%%Determine standard deviation
var_sum_avg = zeros(r,c);
for q = 1:r;
for a = 1:c;
```

```
varsum_avg = (Radius(1,a)-avg_avg_sum(q,1))^2*tPDF_avg(q,a);
var_sum_avg(q,1) = varsum_avg + var_sum_avg(q,1);
end
end
dev_sum_avg = sqrt(var_sum_avg);
% Confidence intervals
fringesets = zeros(set,c,r);
for sets = 1:set;
for row = 1:r;
fringesets(sets,:,row)= nPDF_avg(row,:,sets)/fringe_sum_avg(row,1);
end
end
count_num_fringes = zeros(r,c);
for w = 1:set;
for a = 1:c;
for q = 1:r;
if fringesets(w,a,q)>0;
count_num_fringes(q,a) = 1 + count_num_fringes(q,a);
end
end
end
end
% Variation calculation of fringe sets
```

```
set_var = zeros(set,c,r);
for sets = 1:set;
for row = 1:r;
for clm = 1:c;
if fringesets(sets,clm,row)>0;
set_var(sets,clm,row) = (fringesets(sets,clm,row)- tPDF_avg(row,clm))^2/
count_num_fringes(row,clm);
end
end
end
end
% Summate and square root Variation to get deviation
set_dev = zeros(r,c);
for sets = 1:set;
for clm = 1:c;
for row = 1:r;
set_dev(row,clm) = set_dev(row,clm) + set_var(sets,clm,row);
end
end
end
end
set_dev = sqrt(set_dev);
% Lower confidence interval
lconf = zeros(r,c);
```

```
for row = 1:r;
for clm = 1:c;
if count_num_fringes(row,clm)>1;
lconf(row,clm) = 1.96*set_dev(row,clm)/sqrt(count_num_fringes(row,clm));
end
end
end
% higher confidence interval
hconf = zeros(r,c);
for row = 1:r;
for clm = 1:c;
if count_num_fringes(row,clm)>1;
hconf(row,clm) = 1.96*set_dev(row,clm)/sqrt(count_num_fringes(row,clm));
end
end
end
for row = 1:r;
figure(r + row)
fringe = 0.5-row;
errorbar(Radius,tPDF_avg(row,:),lconf(row,:),hconf(row,:));
title (['Average PDF with error for interference fringe ',num2str(fringe)])
xlabel('Radial positions, cm'), ylabel('Probability')
end
```

X = 0 bias uncertainty quantification code: thoughtexperiment.m

```
%Thought experiment at x = 0
clear, close all, clc
R = 24; %Number of concentric rings
D = 0.046; %Inner diameter of port, m
y = linspace(0,(D/2),R); %y y = 0 at outer diameter of plume, m
%Compute cord length:
for i = 1:max(size(y))
a(i) = 2*sqrt(abs(((D/2)^2)-((D/2)-y(i))^2)); %Cord-length, m
end
%Known Data:
lambda = 514.5; %nm
% lambda = lambda + 50;
lambda = lambda/1e9; %m
T0 = 298; %K, Reference temperature
T1 = 295; %Kelvin, Lower temperature from uncertainty, 22 celcius
T2 = 305; %Kelvin, Upper temperature from uncertainty, 32 celcius
P0 = 101.325; %kPa, Reference pressure
P1 = 96.68; %kPa, Lower pressure from uncertainty
P2 = 101.5; %kPa, Upper pressure from uncertainty
%From Dispersion of Helium article:
Z0He = 1+(0.74062E-8*T0^2-0.061793E-4*T0 + 16.742E-4)*(P0/P0);
%Compressibility factor for Helium, T is in K, P is in atm's
```

```
%From Refrative index of air article:

ao = 1.58123E-6; %K Pa^-1

a1 = 22.9331E-8; %Pa^-1

a2 = 1.1043E-10; %K^-1 Pa^-1

bo = 5.707E-6; %K Pa^-1

b1 = 22.051E-8; %Pa^-1

co = 1.9898E-4; %K Pa^-1

c1 = -2.376E-6; %Pa^-1

d = 1.83E-11; %K^2 Pa^-2

e= 0.765E-8; %K^2 Pa^-2

Tstd = 273; %Kelvin to Celcius

xv = 0; %Mole percent of water in air

Z0air = 1-((P0*1E3)/T0)*(ao+a1*(T0-Tstd)+a2*(T0-Tstd)^2+(bo+b1*(T0- Tstd))*xv

+(co+c1*(T0-Tstd))*xv^2)+((P0*1E3)/T0)^2*(d+e*xv^2);

%Change basis to both 298 K:

nHe = 1.000034985; % @ 273 K

nair = 1.00027861; % @ 288 K

Z273He = 1+(0.74062E-8*(273)^2-0.061793E-4*(273) + 16.742E-4)*(P0/P0);

%Compressibility factor for Helium, T is in K, P is in atm's

rho0rhoHe = ((P0/P0)*T0*Z0He)/((P0/P0)*(273)*Z273He);

ratioHe = rho0rhoHe*(1+(1/6)*(nHe-1)*(1-((P0/P0)*273)/((P0/P0)*T0)));

delnHe = (nHe-1)/ratioHe;

NHe = 1 + delnHe;
```

```
Z288air = 1-((P0*1E3)/288)*(ao+a1*(288-Tstd)+a2*(288-Tstd)^2+(bo+b1*(288-
Tstd))*xv

+(co+c1*(288-Tstd))*xv^2)+((P0*1E3)/288)^2*(d+e*xv^2);

rho0rhoair = ((P0/P0)*T0*Z0air)/((P0/P0)*(288)*Z288air);

ratioair = rho0rhoair*(1+(1/6)*(nair-1)*(1-((P0/P0)*288)/((P0/P0)*T0)));

delnair = (nair-1)/ratioair;

Nair = 1 + delnair;

% NHe = 1.000034985; % @ 273 K

NHe = 1*(NHe-1) + (1-1)*(1.00046-1);

NHe = NHe +1;

% Nair = 1 + 2.81374452437367E-4; % @ 288 K, from

% http://www.strw.leidenuniv.nl/~mathar/progs/prWaterWeb.html

% Nair = 1.00027861; % @ 288 K

%

% NHe = 1 + (NHe-1)*(273/298);

% Nair = 1 + (Nair-1)*(288/298);

% NHe = 1.000034997181250; %Refractive index of helium

dN = NHe - Nair;

% dN = -2.07e-004; %change in refractive index that matches experimental

% data gives fringes down to -18.5

%From error analysis:

% minimum dN when composition is PURE helium and extreme of nm,T,P:

dNmin = -2.4615e-004;

% minimum dN when composition is std helium and extreme of nm,T,P:
```

```
dNmax = -2.1858e-004;

%Analytical solution at a height, x = 0 :

S = (dN)*a/lambda; %Interference fringe order

Smin = (dNmin)*a/lambda;

Smax = (dNmax)*a/lambda;

%Make RHS fringes as well for a pretty picture: (This is just for plotting)

r = D/2-y;

rR = -r;

%Load experimental data:

load Radiusexp

load Svaluesexp

%Preallocate arrays:

Svalues = zeros(min(size(Svaluesexp)),19);

Radius = zeros(min(size(Svaluesexp)),19);

%Extract experimental data from image matrices above:

for i = 1:min(size(Svaluesexp));

[ra c] = find(Svaluesexp(i,:)<0);

Radius(i,11:end) = Radiusexp(1,c)/100;

Svalues(i,11:end) = Svaluesexp(i,c)-10;

Svalues(i,1) = 0;

Svalues(i,2:10) = -1.5:-1:-9.5;

for j = 0:9

Radius(i,j+1) = (-D/2)-(j/10)*((-D/2)-Radius(i,11));
```

```
end
end
%Radius values only for the sets processed: (Helium_01_LHS (1),
%Helium_02_RHS (4), Helium_03_LHS (5), Helium_04_RHS (8))
%Mean and std dev for experimental data:
MeanR = mean(Radius([1,4,5,8],:));
stdevR = std(Radius([1,4,5,8],:));
up = MeanR + stdevR;
down = MeanR - stdevR;
%Use linear interpolation between points:
n = 1;
ri = -0.023:0.001/n:0;
Smean = interp1q(MeanR',Svalues(1,:)',ri');
Sup = interp1q(up',Svalues(1,:)',ri');
Sdown = interp1q(down',Svalues(1,:)',ri');
Sdown(end-1) = -18.5;
Sdown(end) = -18.5;
Smean(end) = -18.5;
upbound = Sup-Smean;
downbound = Smean-Sdown;
figure(1)
% % plot(r,S,'.r',rR,S,'.r')
hold on
```

```
plot(r,Smin,'r',rR,Smin,'r')

plot(r,Smax,'r',rR,Smax,'r')

plot(ri,Sup,'b',ri,Sdown,'b');

plot(-ri,Sup,'b',-ri,Sdown,'b');

% plot(ri,Smean,'.b',-ri,Smean,'.b');

% legend('Ana up','Ana down','Ana up','Ana down','Exp up','Exp up');

xlabel('Radius, m')

ylabel('Inference fringe order')

grid on

hold on

figure(3)

hold on

errorbar(ri',Smean,Smin'-Smean,Sup-Smean,'.k');

errorbar(-ri',Smean,Smin'-Smean,Sup-Smean,'.k');

grid on

xlabel('Radius, m')

ylabel('Inference fringe order')

% %Bias Error determinination:

% biasdown = Smin'-Smean;

% biasup = Smean - Sup;

%For each experimental dataset:

for lr =1:min(size(Svaluesexp))

x = Radius(lr,:)';
```

```
sval = Svalues(i,:)';

ri = (-0.023:0.001/n:0)';

Si(:,lr) = interp1q(x,sval,ri);

ri = fliplr(ri')';

Si(:,lr) = fliplr(Si(:,lr)')';

end

Sdown = fliplr(Sdown')';

Sup = fliplr(Sup')';

%Plot analytical and experimental fringe profiles:

% errorbar(r,S,Smin-S,Smax-S,'k');

% errorbar(rR,S,Smin-S,Smax-S,'k');

%linear interpolations for He1 and He2:

% plot(ri,Si(:,1),'-k',-ri,Si(:,2),'-k',ri,Si(:,3),'-k',-ri,Si(:,4),'-k');

% plot(Radius(1,:),Svalues(1,:),'*b');

% hold on

% plot(-Radius(2,:),Svalues(2,:),'*b');

% hold on

% plot(Radius(3,:),Svalues(3,:),'.g');

% hold on

% plot(-Radius(4,:),Svalues(4,:),'.g');

% hold on

% plot(Radius(5,:),Svalues(5,:),'*r');

% hold on
```

```
% plot(-Radius(6,:),Svalues(6,:),'*r');

% hold on

% plot(Radius(7,:),Svalues(7,:),'.m');

% hold on

% plot(-Radius(8,:),Svalues(8,:),'.m');

% legend('Ana LHS','Ana RHS','He1 LHS','He1 RHS','He2 LHS','He2 RHS','He3
LHS','He3 RHS'

,'He4 LHS','He4 RHS')

%Call abel transformation code for LHS and RHS of experimental data:

Si(1,1:4) = -18.5;

Si(2:3,3:4) = -18.5;

Si(4,3) = -18.5;

% nexpLHS = linabel(Si(:,1),ri);

% nexpRHS = linabel(Si(:,2),ri);

nexpLHS = againabel(Sup,ri,n,Nair);

nexpRHS = againabel(Sdown,ri,n,Nair);

% nexpLHS2 = againabel(Si(:,3),ri,n,Nair);

% nexpRHS2 = againabel(Si(:,4),ri,n,Nair);

SRLHS = -1*y;

%Call abel transformation code for the analytical data:

S = fliplr(S);

Smin = fliplr(Smin);

Smax = fliplr(Smax);
```

```
n = 1;
nsimLHS = againabel(S,SRLHS,n,Nair);
nsimmin = againabel(Smin,SRLHS,n,Nair);
nsimmax = againabel(Smax,SRLHS,n,Nair);
% nsimLHS = linabel(S,SRLHS);
%Plot RI from abel inversion code:
figure(2)
plot(ri(1:(end-1)),nexpLHS(1:(end-1)),'b',ri(1:(end-1)),nexpRHS(1:(end- 1)),'b');
hold on
% plot(SRLHS,nsimLHS,'k');
plot(SRLHS(1:(end-1)),nsimmin(1:(end-1)),'k');
plot(SRLHS(1:(end-1)),nsimmax(1:(end-1)),'k');
xlabel('Radius, m')
ylabel('Refractive index')
% legend('exp up','exp down','Ana Soln');
grid on
```

Quantify analytical solution range: TandPversusRI.m

```
clear all, close all, clc
%Refractive index change with Temperature and Pressure
T0 = 298; %K, Reference temperature
T1 = 295; %Kelvin, Lower temperature from uncertainty, 22 celcius
T2 = 305; %Kelvin, Upper temperature from uncertainty, 32 celcius
P0 = 101.325; %kPa, Reference pressure
P1 = 96.68; %kPa, Lower pressure from uncertainty
P2 = 101.5; %kPa, Upper pressure from uncertainty
%From Dispersion of Helium article:
Z0He = 1+(0.74062E-8*T0^2-0.061793E-4*T0 + 16.742E-4)*(P0/P0);
%Compressibility factor for Helium, T is in K, P is in atm's
%From Refrative index of air article:
ao = 1.58123E-6; %K Pa^-1
a1 = 22.9331E-8; %Pa^-1
a2 = 1.1043E-10; %K^-1 Pa^-1
bo = 5.707E-6; %K Pa^-1
b1 = 22.051E-8; %Pa^-1
co = 1.9898E-4; %K Pa^-1
c1 = -2.376E-6; %Pa^-1
d = 1.83E-11; %K^2 Pa^-2
e= 0.765E-8; %K^2 Pa^-2
Tstd = 273; %Kelvin to Celcius
```

```
xv = 0; %Mole percent of water in air

Z0air = 1-((P0*1E3)/T0)*(ao+a1*(T0-Tstd)+a2*(T0-Tstd)^2+(bo+b1*(T0- Tstd))*xv

+(co+c1*(T0-Tstd))*xv^2)+((P0*1E3)/T0)^2*(d+e*xv^2);

%If we wanted to account for 68% humidity, this increases Nair which is

%bad:

% http://www.climatetemp.info/germany/stuttgart.html

% nair = 1 + 2.81374452437367E-4; % @ 288 K, from

% http://www.strw.leidenuniv.nl/~mathar/progs/prWaterWeb.html

%Change basis of Helium and Air refractive indices to 298 K:

nHe = 1.000034985; % @ 273 K

nair = 1.00027861; % @ 288 K

Z273He = 1+(0.74062E-8*(273)^2-0.061793E-4*(273) + 16.742E-4)*(P0/P0);

%Compressibility factor for Helium, T is in K, P is in atm's

rho0rhoHe = ((P0/P0)*T0*Z0He)/((P0/P0)*(273)*Z273He);

ratioHe = rho0rhoHe*(1+(1/6)*(nHe-1)*(1-((P0/P0)*273)/((P0/P0)*T0)));

delnHe = (nHe-1)/ratioHe;

nHe = 1 + delnHe;

Z288air = 1-((P0*1E3)/288)*(ao+a1*(288-Tstd)+a2*(288-Tstd)^2+(bo+b1*(288-
Tstd))*xv

+(co+c1*(288-Tstd))*xv^2)+((P0*1E3)/288)^2*(d+e*xv^2);

rho0rhoair = ((P0/P0)*T0*Z0air)/((P0/P0)*(288)*Z288air);

ratioair = rho0rhoair*(1+(1/6)*(nair-1)*(1-((P0/P0)*288)/((P0/P0)*T0)));

delnair = (nair-1)/ratioair;

nair = 1 + delnair;
```

```
% nair = 1 + (nair - 1)*((P0*1E3)/96095.43)*((1+1E-8*(0.601-0.00972*(T0-
273))*P0*1E3)

/(1+0.0036610*(T0-273)));

%Change composition of helium to partial of methane:

Pure = 1;

UHP = 0.9995;

Standard = 0.995;

% Arbh = 0.9651;

Arbl = 0.9087;

comp = Standard;

nHe = comp*(nHe-1) + (1-comp)*(1.0004478-1);

nHe = nHe + 1;

%Set values for later:

RI1 = nHe;

dNstd = nHe-nair;

% %Add in Joule-Thompson coefficient to change lower temperature bound:

Ptank = 1200; %Pisg, pressure of Helium tank:

Patm = 14.7; %Psia

Ptank = Ptank + Patm; %Psia

Ptank = Ptank/Patm; %bar

JT_coeff = -0.07; %K/bar, from wikipedia article for Joule-Thompson effect

DelT = JT_coeff*(Ptank-1); % 1 bar is atmospheric

T1 = T1+DelT; %K, new lower temperature for uncertainty
```

```
n = 100;

T = linspace(T1,T2,n);

P = linspace(P1,P2,n);

delnair = zeros(n,n);

delnHe = zeros(n,n);

RI2 = zeros(n,n);

ZHe = zeros(n,n);

Zair = zeros(n,n);

rho0rhoHe = zeros(n,n);

ratioHe = zeros(n,n);

[TT PP] = meshgrid(T,P);

for i = 1:n

for j = 1:n

ZHe(i,j) = 1+(0.74062E-8*TT(i,j)^2-0.061793E-4*TT(i,j) + 16.742E- 4)*(PP(i,j)/P0);

%Compressibility factor for Helium, T is in K, P is in atm's

rho0rhoHe(i,j) = ((P0/P0)*TT(i,j)*ZHe(i,j))/((PP(i,j)/P0)*T0*Z0He);

ratioHe(i,j) = rho0rhoHe(i,j)*(1+(1/6)*(nHe-1)*(1- ((PP(i,j)/P0)*T0)/((P0/P0)*TT(i,j))));

delnHe(i,j) = (nHe-1)/ratioHe(i,j);

Zair(i,j)=1-((PP(i,j)*1E3)/TT(i,j))*(ao+a1*(TT(i,j)-Tstd)+a2*(TT(i,j)-
Tstd)^2+(bo+b1*(TT(i,j)-Tstd))*xv

+(co+c1*(TT(i,j)-Tstd))*xv^2)+((PP(i,j)*1E3) /TT(i,j))^2*(d+e*xv^2);

rho0rhoair(i,j) = ((P0/P0)*TT(i,j)*Zair(i,j))/((PP(i,j)/P0)*T0*Z0air);

ratioair(i,j) = rho0rhoair(i,j)*(1+(1/6)*(nair-1)*(1- ((PP(i,j)/P0)*T0)/((P0/P0)*TT(i,j))));

delnair(i,j) = (nair-1)/ratioair(i,j);
```

```
RI2(i,j) = delnHe(i,j)-delnair(i,j); %New refractive index of Helium

end

end

%Just effect of temperature and pressure on difference in refractive index:

figure(1), contourf(TT,PP,RI2);

grid on

xlabel('Temperature, K')

ylabel('Pressure, kPa')

zlabel('Refractive index of helium')

wvlngth = linspace((0.5145-0.050), (0.5145+0.050), n);

delnHe_wvlngth = zeros(1,n);

delnair_wvlngth = zeros(1,n);

delRI2_wvlngth = zeros(1,n);

rho0rhoHewvlngth = zeros(1,n);

ratioHewvlngth = zeros(1,n);

rho0rhoairwvlngth = zeros(1,n);

ratioairwvlngth = zeros(1,n);

for i = 1:n;

delnHe_wvlngth(i) = (0.01470091/(423.98-wvlngth(i)^-2));

rho0rhoHewvlngth(i) = ((P0/P0)*T0*Z0He)/((P0/P0)*(273)*Z273He);

ratioHewvlngth(i) = rho0rhoHewvlngth(i)*(1+(1/6)*(delnHe_wvlngth(i))*(1-
((P0/P0)*273)

/((P0/P0)*T0)));
```

```
delnHe_wvlngth(i) = (delnHe_wvlngth(i))/ratioHewvlngth(i);

delnHe_wvlngth(i) = comp*delnHe_wvlngth(i) + (1-comp)*(1.0004478-1);

%Add in impurity of methane

delnair_wvlngth(i) = 5792105E-8/(238.0185-wvlngth(i)^-2)+167917E- 8/(57.362-
wvlngth(i)^-2);

rho0rhoairwvlngth(i) = ((P0/P0)*T0*Z0air)/((P0/P0)*(288)*Z288air);

ratioairwvlngth(i) = rho0rhoairwvlngth(i)*(1+(1/6)*(delnair_wvlngth(i))*(1-
((P0/P0)*288)

/((P0/P0)*T0)));

delnair_wvlngth(i) = (delnair_wvlngth(i))/ratioairwvlngth(i);

delRI2_wvlngth(i) = delnHe_wvlngth(i)-delnair_wvlngth(i);

end

%Just effect of wavelength on difference in refractive index:

figure(2), plot(wvlngth,delRI2_wvlngth);

grid on

xlabel('Wavelength, micrometers')

ylabel('Change in refractive index')

delnair = zeros(n,n,n);

delnHe = zeros(n,n,n);

delRI2_all = zeros(n,n,n);

for k = 1:n

for i = 1:n

for j = 1:n

ZHe(i,j,k) = 1+(0.74062E-8*TT(i,j)^2-0.061793E-4*TT(i,j) + 16.742E-4)*(PP(i,j)/P0);
```

```
%Compressibility factor for Helium, T is in K, P is in atm's

rho0rhoHe(i,j,k) = ((P0/P0)*TT(i,j)*ZHe(i,j,k))/((PP(i,j)/P0)*T0*Z0He);

ratioHe(i,j,k) = rho0rhoHe(i,j,k)*(1+(1/6)*(delnHe_wvlngth(k))*(1- ((PP(i,j)/P0)*T0)

/((P0/P0)*TT(i,j))));

delnHe(i,j,k) = (delnHe_wvlngth(k))/ratioHe(i,j);

Zair(i,j,k) = 1-((PP(i,j)*1E3)/TT(i,j))*(ao+a1*(TT(i,j)- Tstd)

+a2*(TT(i,j)-Tstd)^2+(bo+b1*(TT(i,j)-Tstd))*xv+(co+c1*(TT(i,j)- Tstd))*xv^2)+

((PP(i,j)*1E3)/TT(i,j))^2*(d+e*xv^2);

rho0rhoair(i,j,k) = ((P0/P0)*TT(i,j)*Zair(i,j,k))/((PP(i,j)/P0)*T0*Z0air);

ratioair(i,j,k) = rho0rhoair(i,j,k)*(1+(1/6)*(delnair_wvlngth(k))*(1-
((PP(i,j)/P0)*T0)/((P0/P0)*TT(i,j))));

delnair(i,j,k) = (delnair_wvlngth(k))/ratioair(i,j);

delRI2_all(i,j,k) = delnHe(i,j,k)-delnair(i,j,k);

end

end

end

max_diff = max(max(max(delRI2_all)))

min_diff = min(min(min(delRI2_all)))

%T,P, and lambda effect on delRI:

figure(3), subplot(1,3,1),contourf(TT,PP,delRI2_all(:,:,1)); title('464.5 nm')

grid on

xlabel('Temperature, K')

ylabel('Pressure, kPa')

zlabel('Refractive index of helium')
```

```
subplot(1,3,2),contourf(TT,PP,delRI2_all(:,:,n/2));title('514.5 nm')
grid on
xlabel('Temperature, K')
ylabel('Pressure, kPa')
zlabel('Refractive index of helium')
subplot(1,3,3),contourf(TT,PP,delRI2_all(:,:,n)); title('564.5 nm')
grid on
xlabel('Temperature, K')
ylabel('Pressure, kPa')
zlabel('Refractive index of helium')
% for i = 1:n
% contourf(TT,PP,delRI2_all(:,:,i))
% grid on
% xlabel('Temperature, K')
% ylabel('Pressure, kPa')
% zlabel('Refractive index of helium')
% M(i) = getframe;
% end
%
% figure(3), movie(M,1)
```

Output final experimental graph with means and error bars: RI_3and5.m

```
%Determine Refractive index for x = 3 and 5 cm data:
clear, close all, clc
Radius_3cm = [-0.0169873595505618,-0.0139606741573034,-0.0119077715355805,
-0.00993913857677903,-0.00739700374531835,-0.00823843077509046,
- 0.00756454697585465,-0.00642098192476795,-0.00566011235955056,
- 0.00479040622299049,-0.00366221910112360];
Svalues_3cm = [-0.5:-1:-10.5];
Radius_5cm = [-0.0175367251684162,-0.0144962140537366,-0.0125608533295716,
-0.0104629208844415,-0.00783884516246559,-0.00688616988873948,
-0.00606902807483909,-0.00521998145119886,-0.00413454634600022,
- 0.00301172679103489,-0.00121166920743668];
Svalues_5cm = [-0.5:-1:-10.5];
%Bias Error:
biasdown = [0;-3.35852360061699;-3.35560068069613;-2.68705140314667;
-1.80784428221900;-2.12293089664228;-2.45085108564419;-2.67697407873482;-
2.80306435786257;-2.85871451613610;-2.93095970010844;-3.06469889917987;-
3.23965069221417;-3.35308870952348;-3.47019215366394;-3.53995666109471;-
3.55919509718348;-3.62732153872487;-3.69228985154755;-3.71249472304242;-
3.68911139177003;-3.62301878845283;-3.51482677370938;-3.50758017492711];
biasup = [0;-0.251899257221349;-0.503798514442692;-0.755697771664035;
-0.861042693885816;-0.435656076289984;-0.407372173737814;-0.451826778721610;-
```

```
0.552260954545506;-0.617897919846335;-0.602105929175471;-0.529701541725208;-
0.404599362144625;-0.353962018796718;-0.251321122517965;-0.227253806138673;-
0.203186489759389;-0.100845200205431;-0.110599951099847;-0.120354701994259;-
0.130109452888671;-0.139864203783084;-0.149618954677493;-0.0166878542529609];
biasdown = fliplr(biasdown')';
biasup = fliplr(biasup')';
biasrad = (-0.023:0.001:0)';
%Image processing uncertainty:
load SA_Helium_01_3cm
load SA_Helium_04_3cm
load SA_Helium_01_5cm
load SA_Helium_04_5cm
%Change to meters from pixels
% 1:2 - Helium_01, 3:4 - Helium_04
err3cm(1:2,:) = SA_Helium_01_3cm(:,1:max(size(Radius_3cm)))/(178*2*100);
err3cm(3:4,:) = SA_Helium_04_3cm(:,1:max(size(Radius_3cm)))/(178*2*100);
err5cm(1:2,:) = SA_Helium_01_5cm(:,1:max(size(Radius_5cm)))/(178*2*100);
err5cm(3:4,:) = SA_Helium_04_5cm(:,1:max(size(Radius_5cm)))/(178*2*100);
%Determine average: row 1 & 3- up (right), row 2 & 4 - down (left)
avgerr3cm(1,:) = (err3cm(1,:)+err3cm(3,:))/2;
avgerr3cm(2,:) = -(err3cm(2,:)+err3cm(4,:))/2;
avgerr5cm(1,:) = (err5cm(1,:)+err5cm(3,:))/2;
avgerr5cm(2,:) = -(err5cm(2,:)+err5cm(4,:))/2;
```

```
%Variability error:
load stddev3cm
load stddev5cm
%Total experimental error:
for i = 1:2
avgerr3cm(i,:) = avgerr3cm(i,:) + (stddev3cm(1:max(size(avgerr3cm)))/100)';
avgerr5cm(i,:) = avgerr5cm(i,:) + (stddev5cm(1:max(size(avgerr5cm)))/100)';
end
figure(1)
% plot(Radius_3cm, Svalues_3cm,'.k',Radius_5cm, Svalues_5cm,'*b');
herrorbar(Radius_3cm,Svalues_3cm,avgerr3cm(1,:),avgerr3cm(2,:),'.k');
hold on
xlabel('Radius, m')
ylabel('Inference fringe order')
grid on
figure(2)
herrorbar(Radius_5cm,Svalues_5cm,avgerr5cm(1,:),avgerr5cm(2,:),'.b');
hold on
xlabel('Radius, m')
ylabel('Inference fringe order')
grid on
% legend('Exp 3cm','Exp 5cm','Lin 3cm','Lin 5cm');
% title('Interpolation on x = 3 and 5 cm inference data');
```

```
%Interpolate uncertainty to get error in fringe value:

S5cmup = interp1((Radius_5cm+avgerr5cm(1,:))',Svalues_5cm',Radius_5cm','linear');

S5cmdown = interp1((Radius_5cm- avgerr5cm(2,:))',Svalues_5cm',Radius_5cm','linear');

S3cmup = interp1((Radius_3cm+avgerr3cm(1,:))',Svalues_3cm',Radius_3cm','linear');

S3cmdown = interp1((Radius_3cm- avgerr3cm(2,:))',Svalues_3cm',Radius_3cm','linear');

%Clear NaN to zeros:

S5cmup(1:2) = 0;

S5cmdown(11) = 0;

S3cmup(1:2) = 0;

S3cmdown(8:11) = 0;

%Interpolate bias to get erro in fringe value:

bias5cmup = interp1(biasrad,biasup,Radius_5cm','linear');

bias5cmdown = interp1(biasrad,biasdown,Radius_5cm','linear');

bias3cmup = interp1(biasrad,biasup,Radius_3cm','linear');

bias3cmdown = interp1(biasrad,biasdown,Radius_3cm','linear');

figure(3), hold on

errorbar(Radius_3cm,Svalues_3cm,S3cmdown-
Svalues_3cm'+bias3cmdown,Svalues_3cm'-

S3cmup-bias3cmup,'.k');

xlabel('Radius, m')

ylabel('Inference fringe order')

grid on

figure(4), hold on
```

```
errorbar(Radius_5cm,Svalues_5cm,S5cmdown-
Svalues_5cm'+bias5cmdown,Svalues_5cm'-

S5cmup-bias5cmup,'.b');

xlabel('Radius, m')

ylabel('Inference fringe order')

grid on
```

APPENDIX C

INTERFERENCE FRINGE PDFS

x = 3 cm

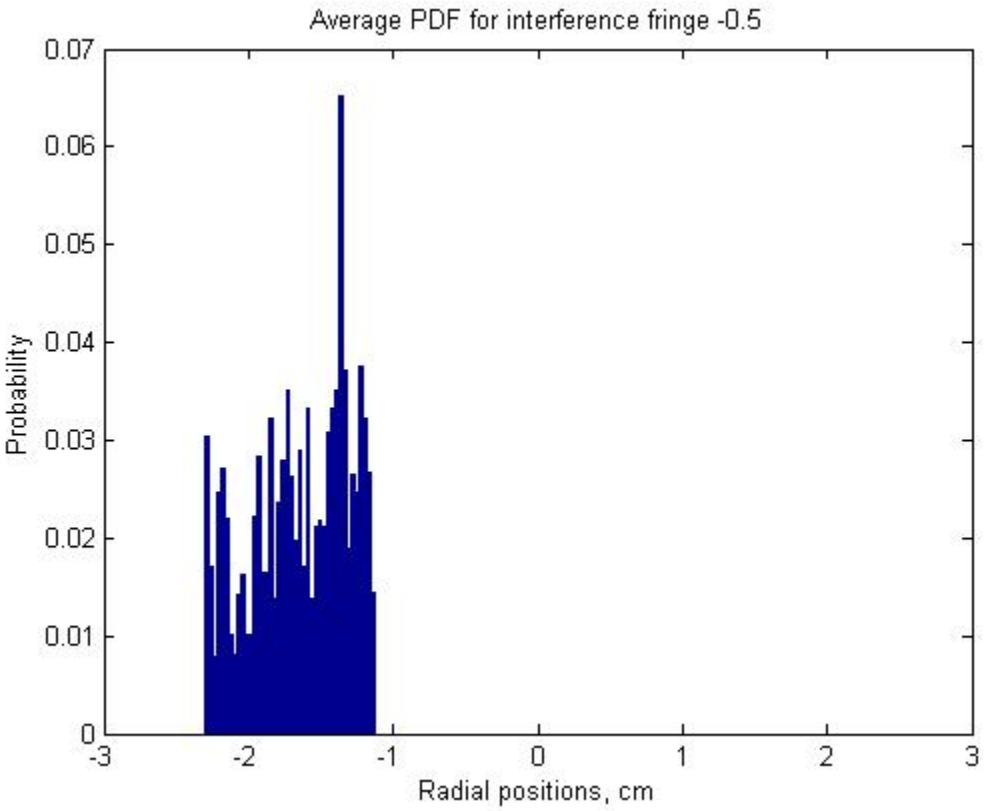


Figure 32: PDF for fringe -0.5 at x = 3 cm

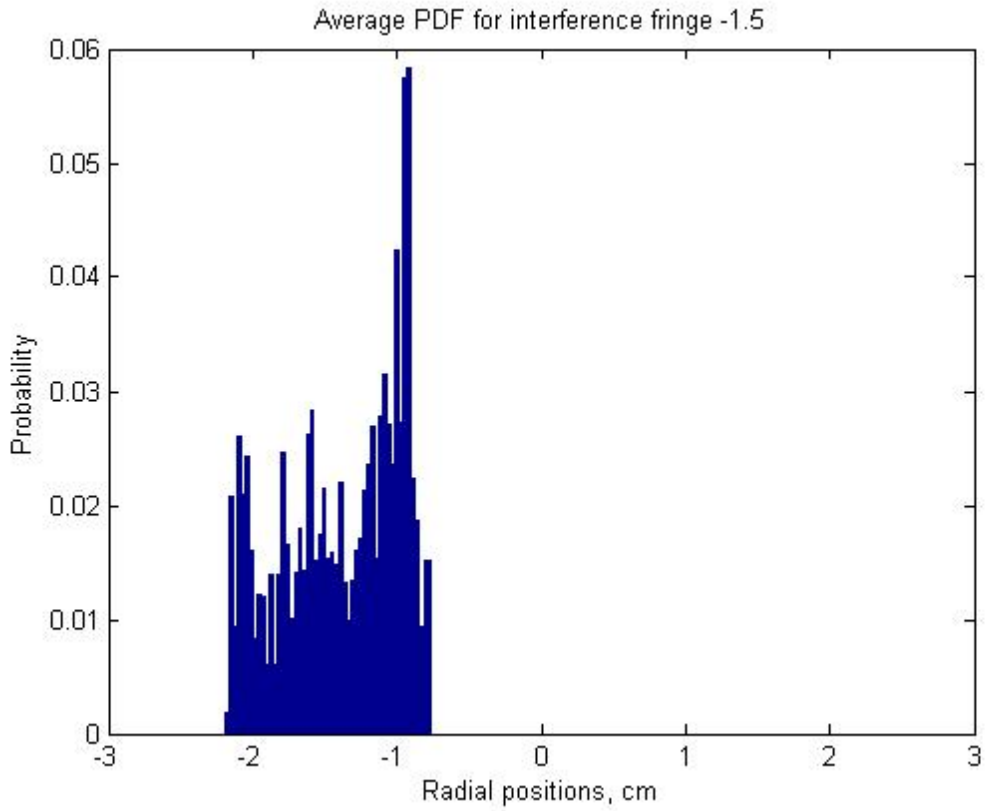


Figure 33: PDF for fringe -1.5 at x = 3 cm

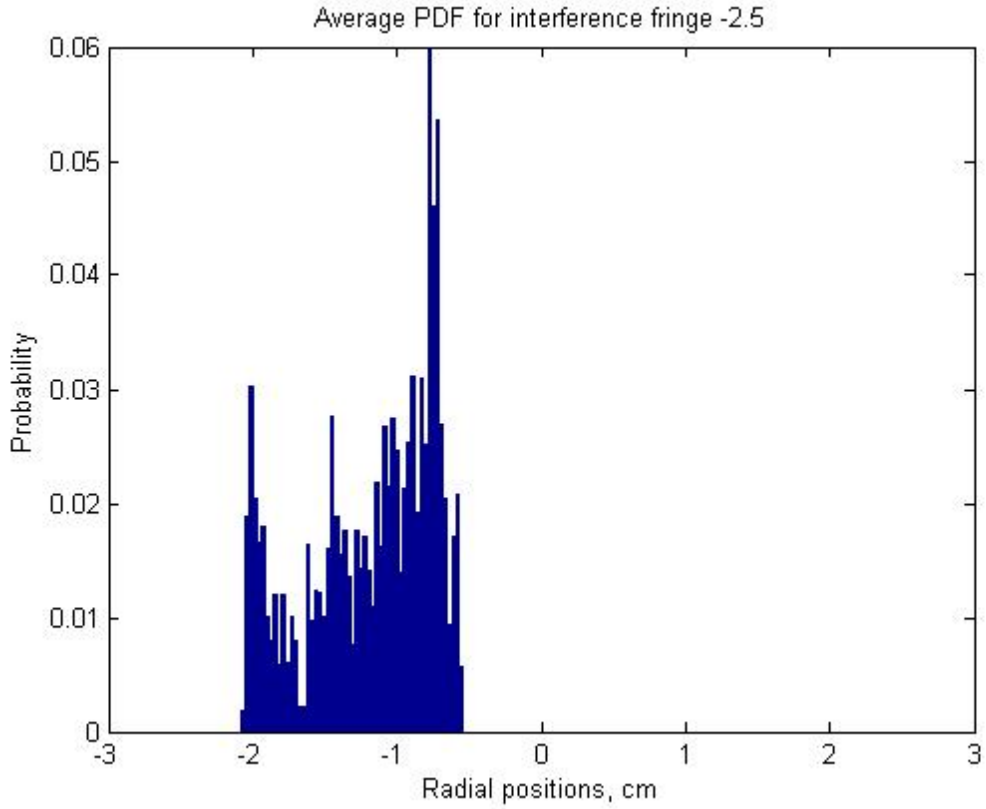


Figure 34: PDF for fringe -2.5 at x = 3 cm

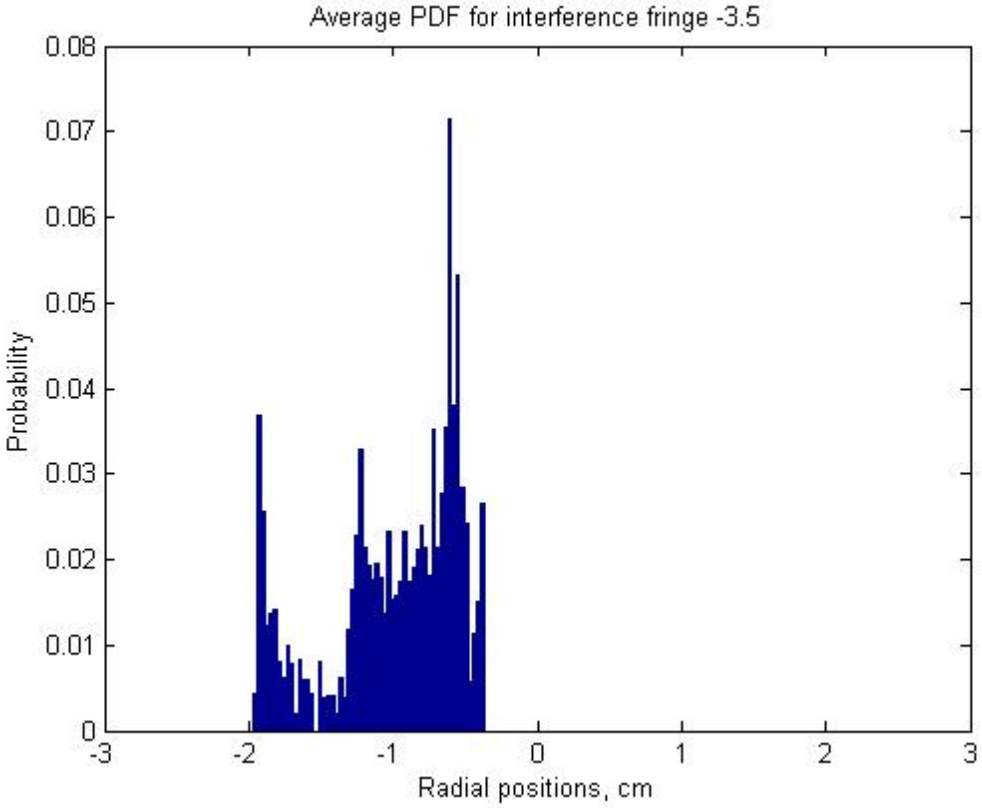


Figure 35: PDF for fringe -3.5 at x = 3 cm

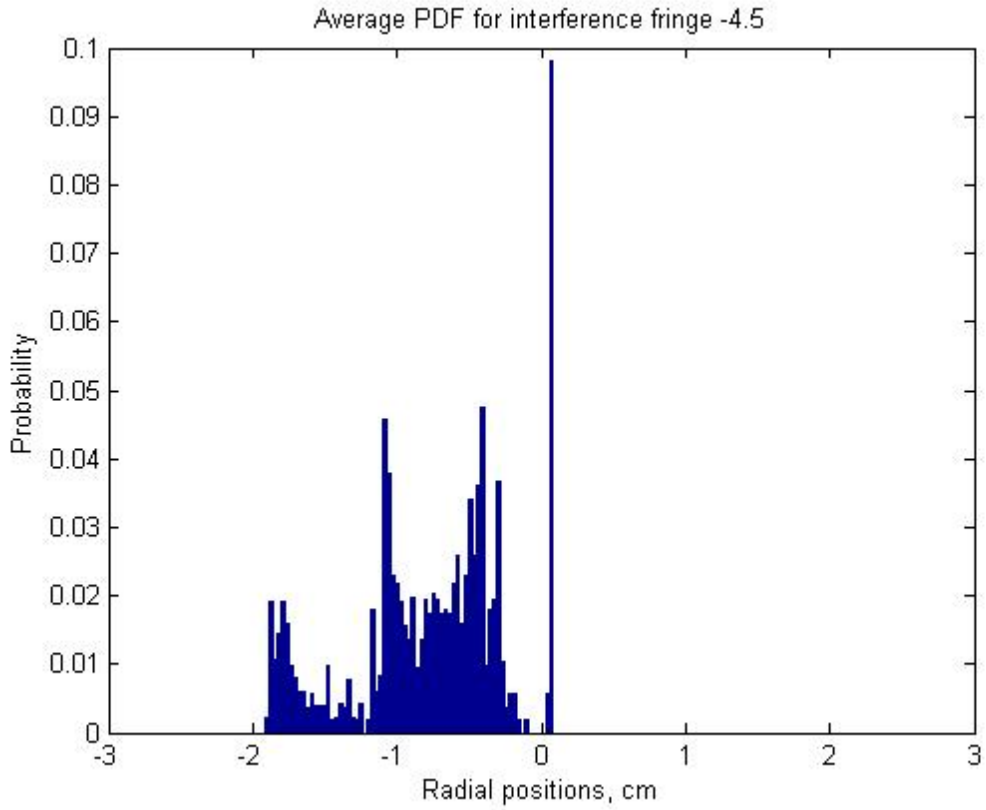
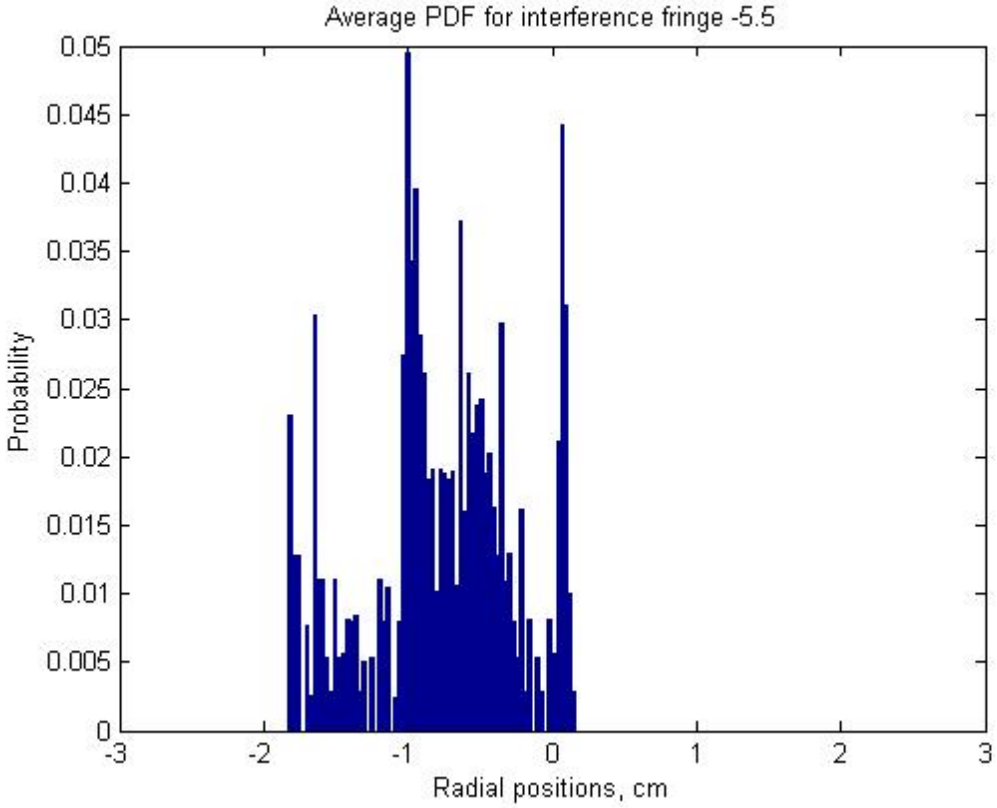


Figure 36: PDF for fringe -4.5 at x = 3 cm



`Figure 37: PDF for fringe -5.5 at x = 3 cm

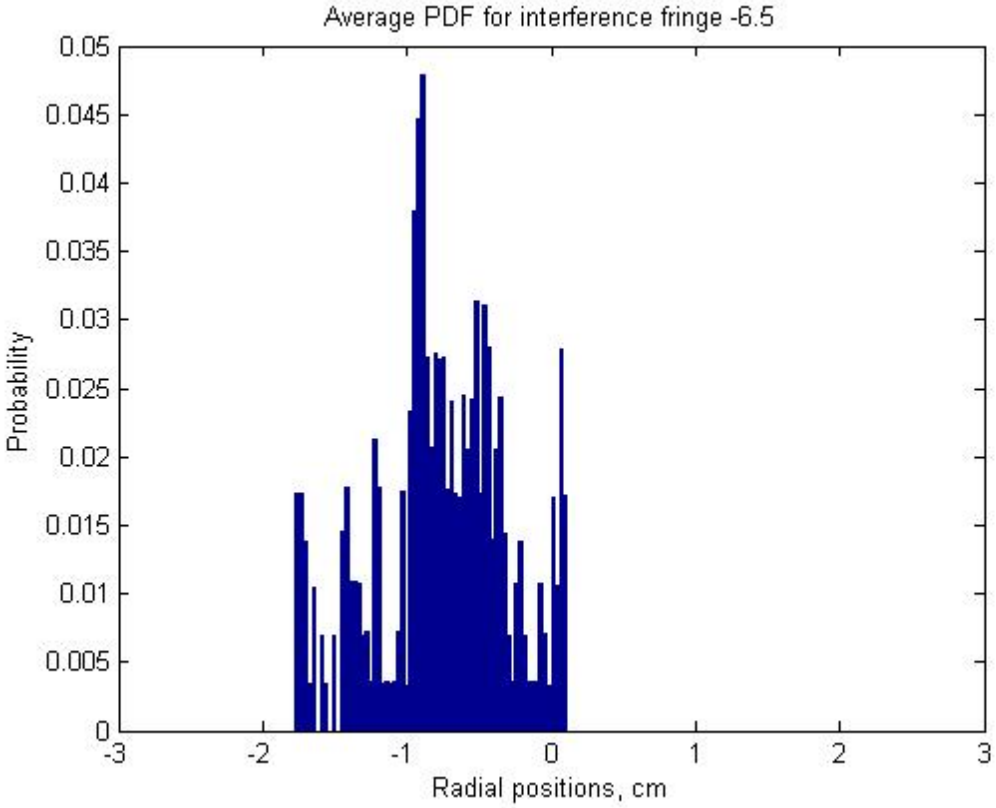


Figure 38: PDF for fringe -6.5 at x = 3 cm

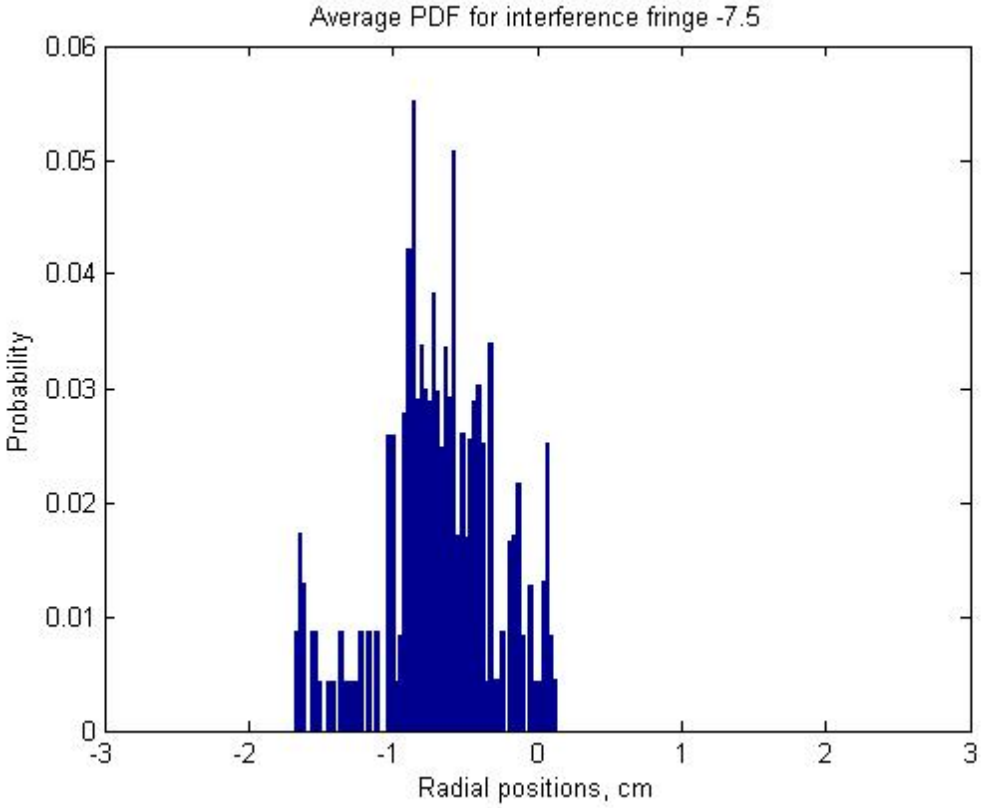


Figure 39: PDF for fringe -7.5 at x = 3 cm

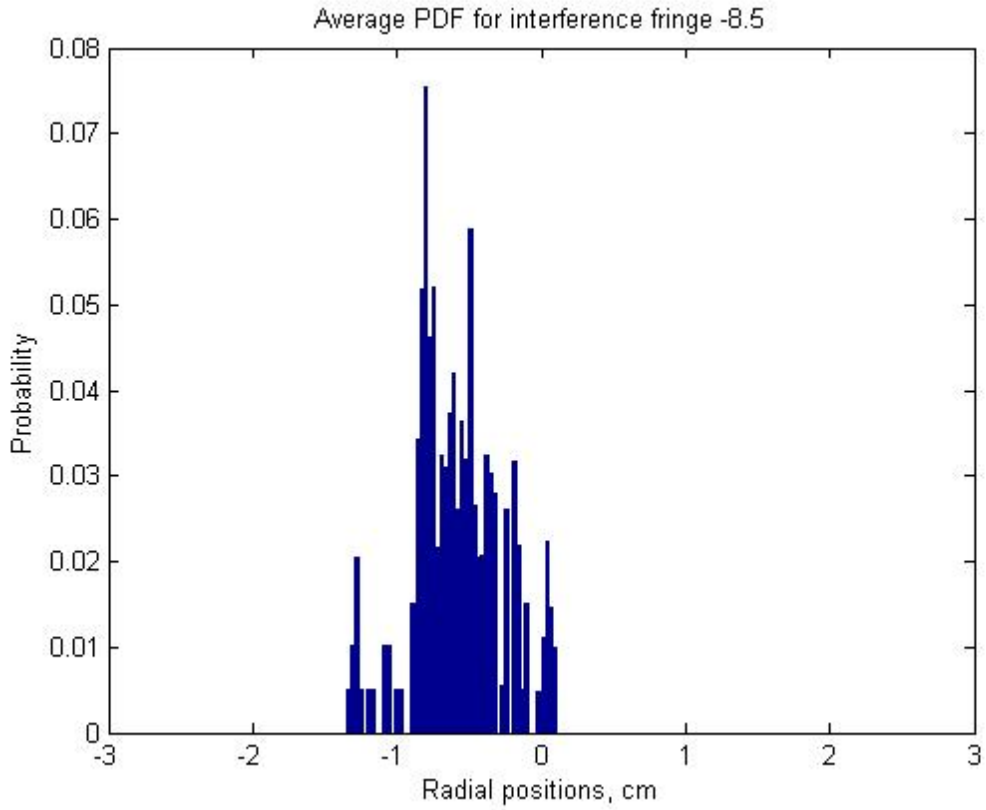


Figure 40: PDF for fringe -8.5 at x = 3 cm

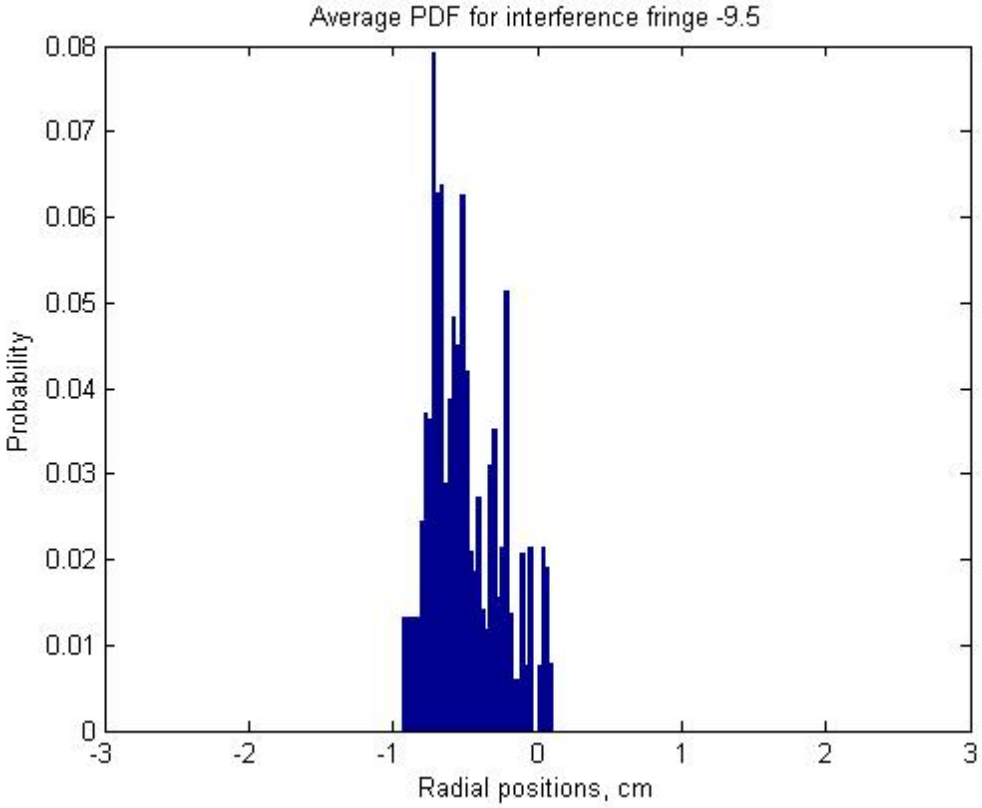


Figure 41: PDF for fringe -9.5 at x = 3 cm

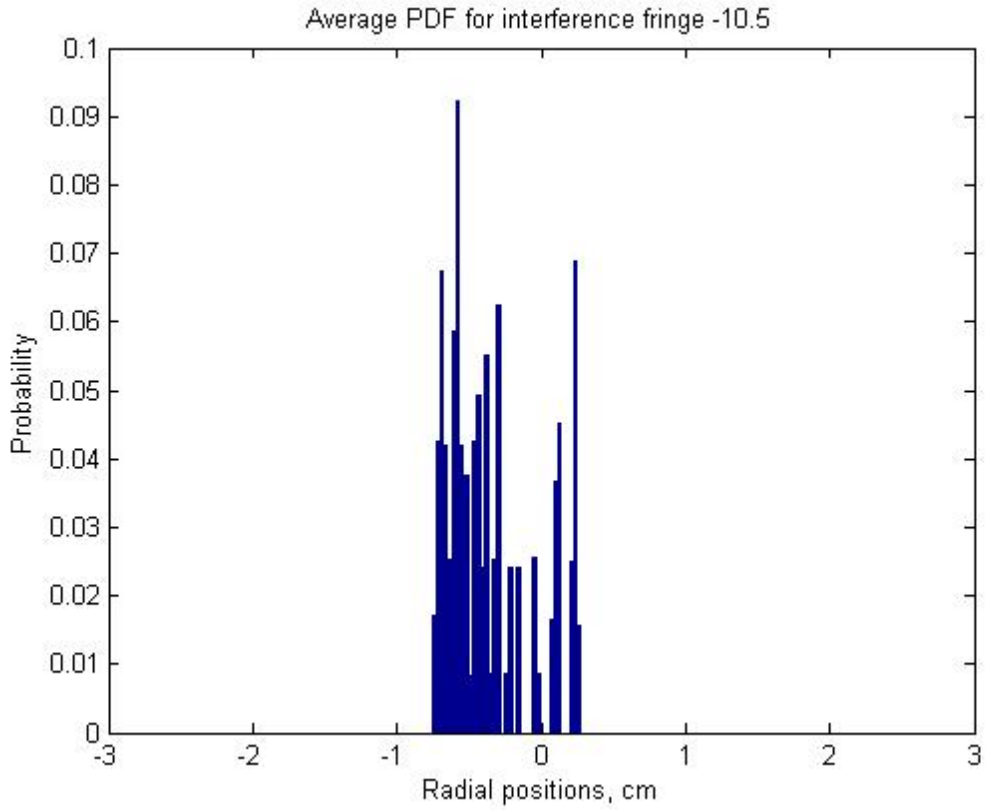


Figure 42: PDF for fringe -10.5 at x = 3 cm

x = 5 cm

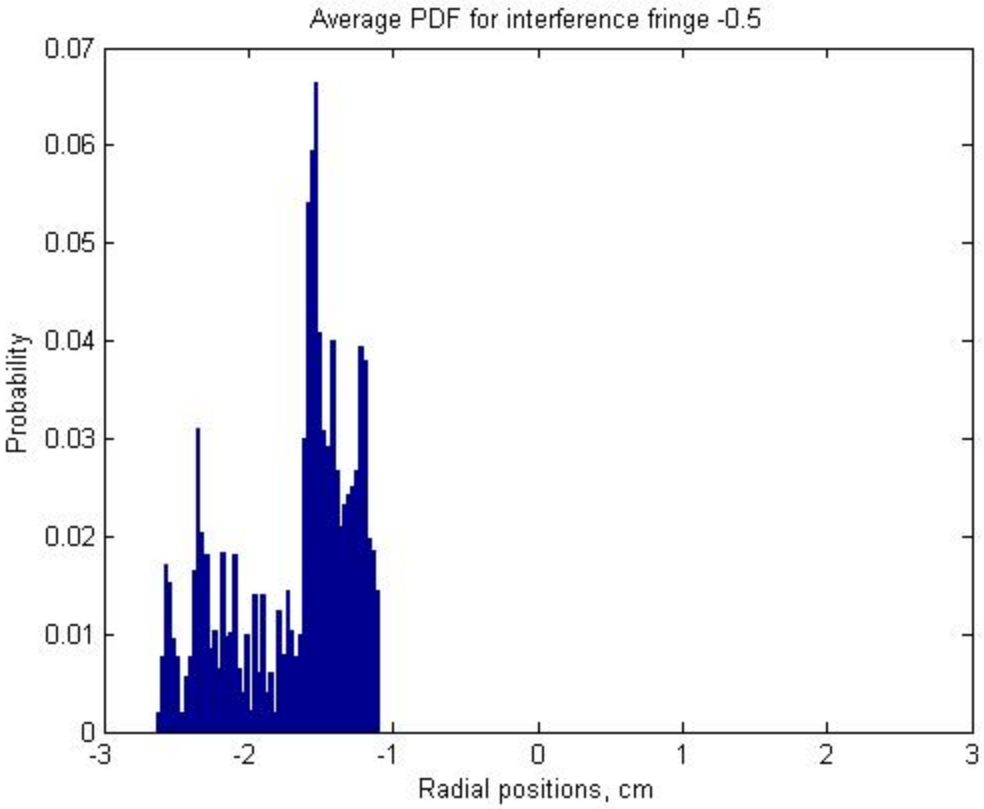


Figure 43: PDF for fringe -0.5 at x = 5 cm

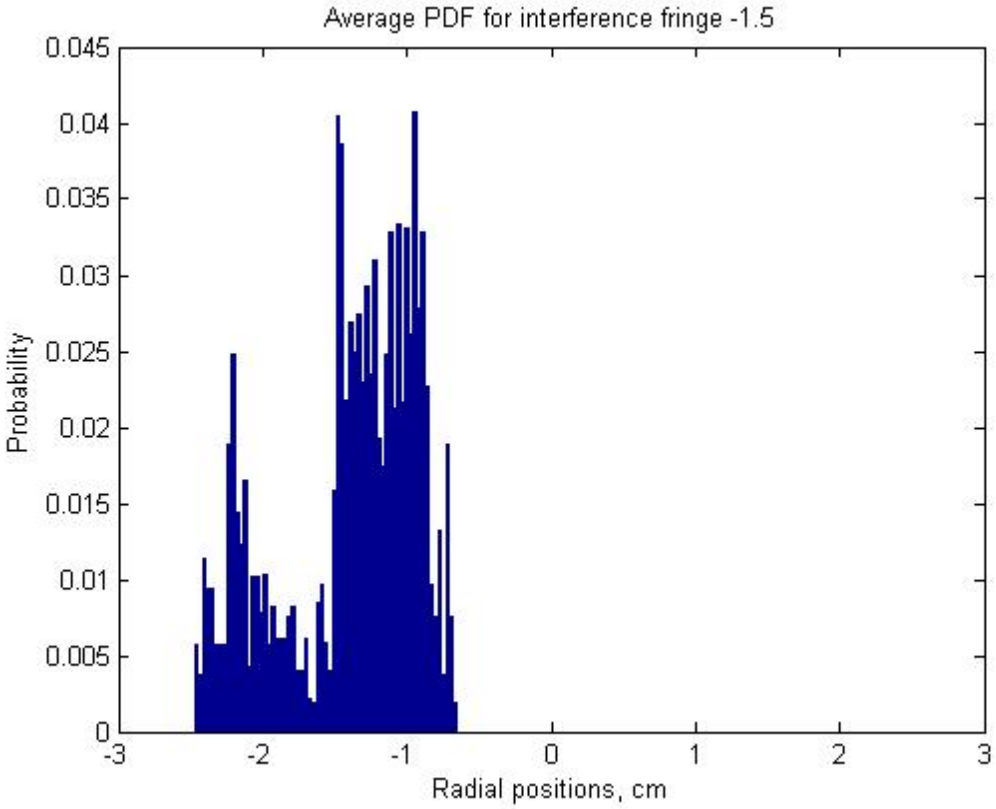


Figure 44: PDF for fringe -1.5 at x = 5 cm

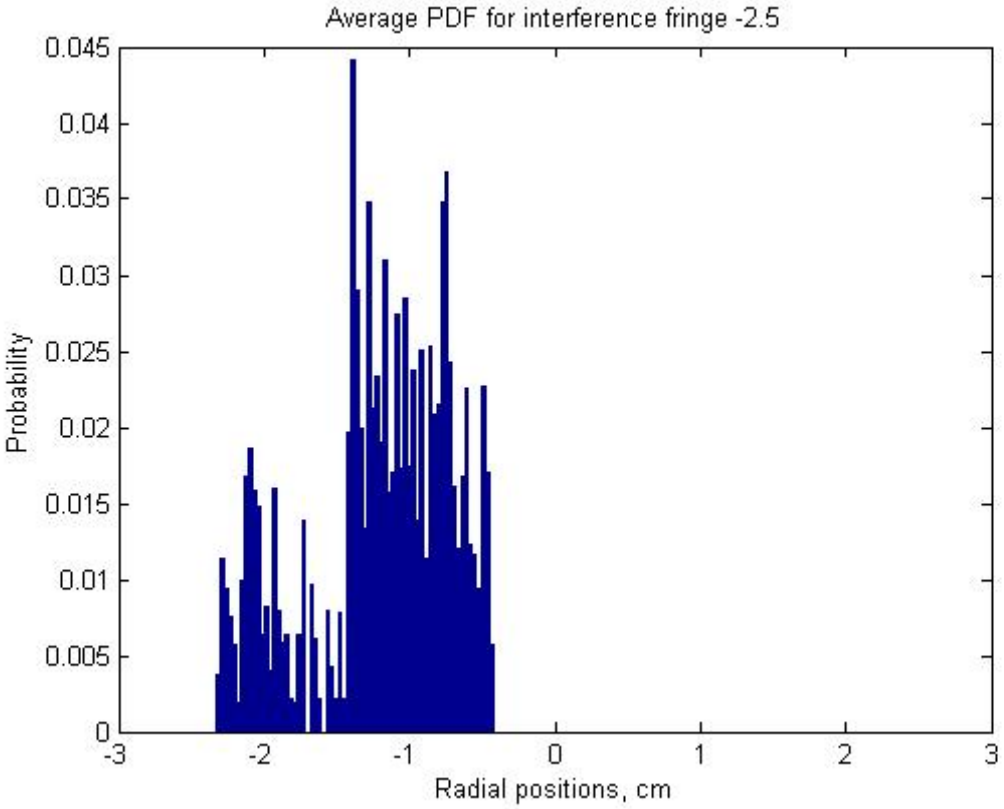


Figure 45: PDF for fringe -2.5 at x = 5 cm

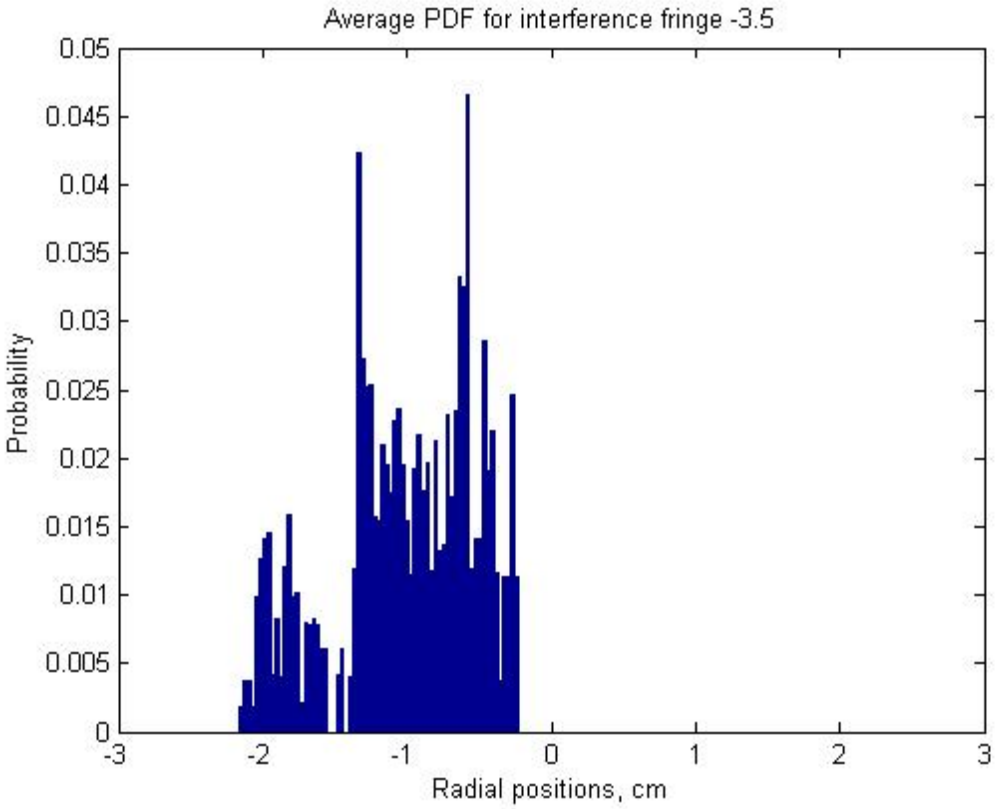


Figure 46: PDF for fringe -3.5 at x = 5 cm

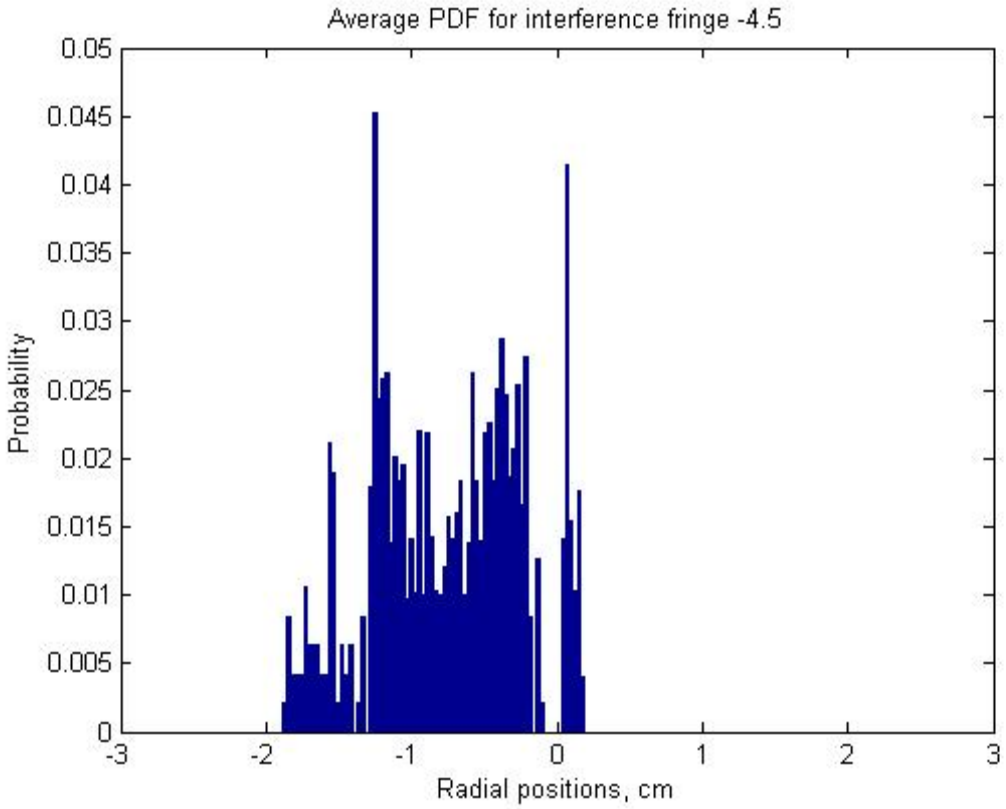


Figure 47: PDF for fringe -4.5 at x = 5 cm

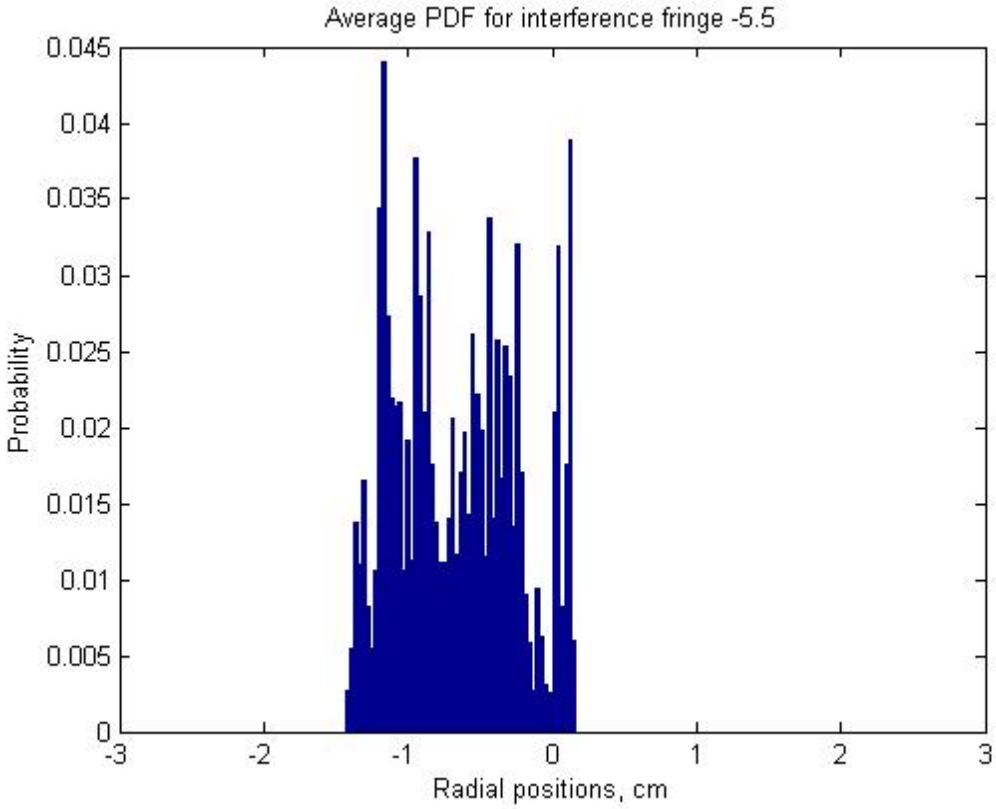


Figure 48: PDF for fringe -5.5 at x = 5 cm

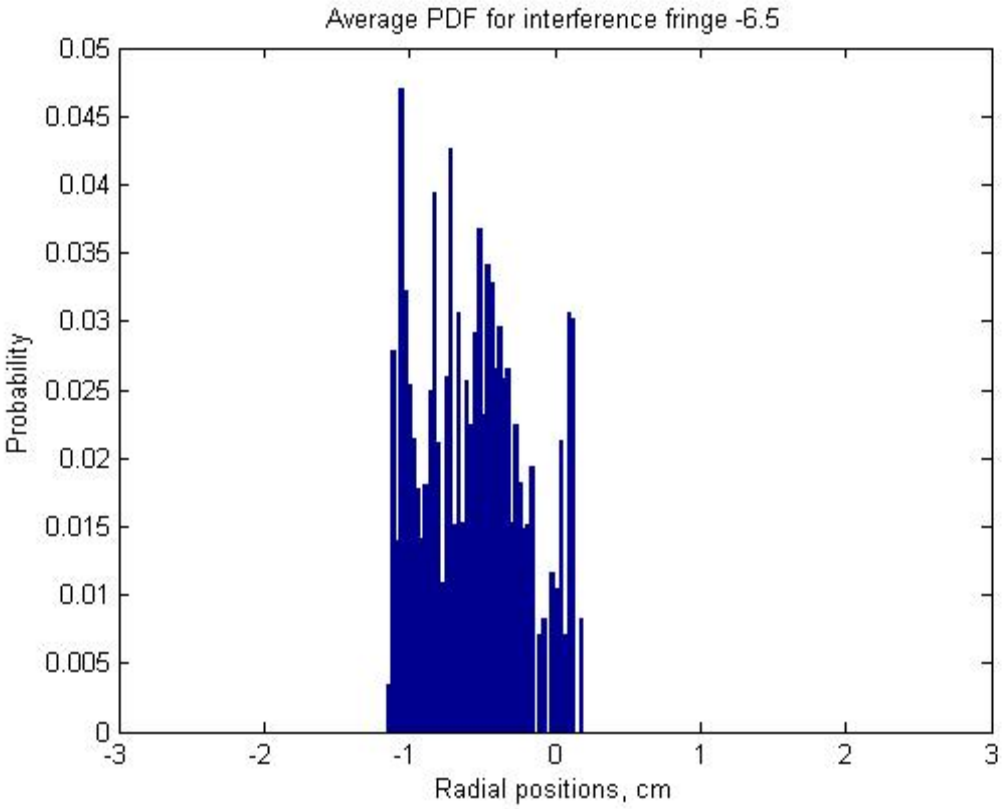


Figure 49: PDF for fringe -6.5 at x = 5 cm

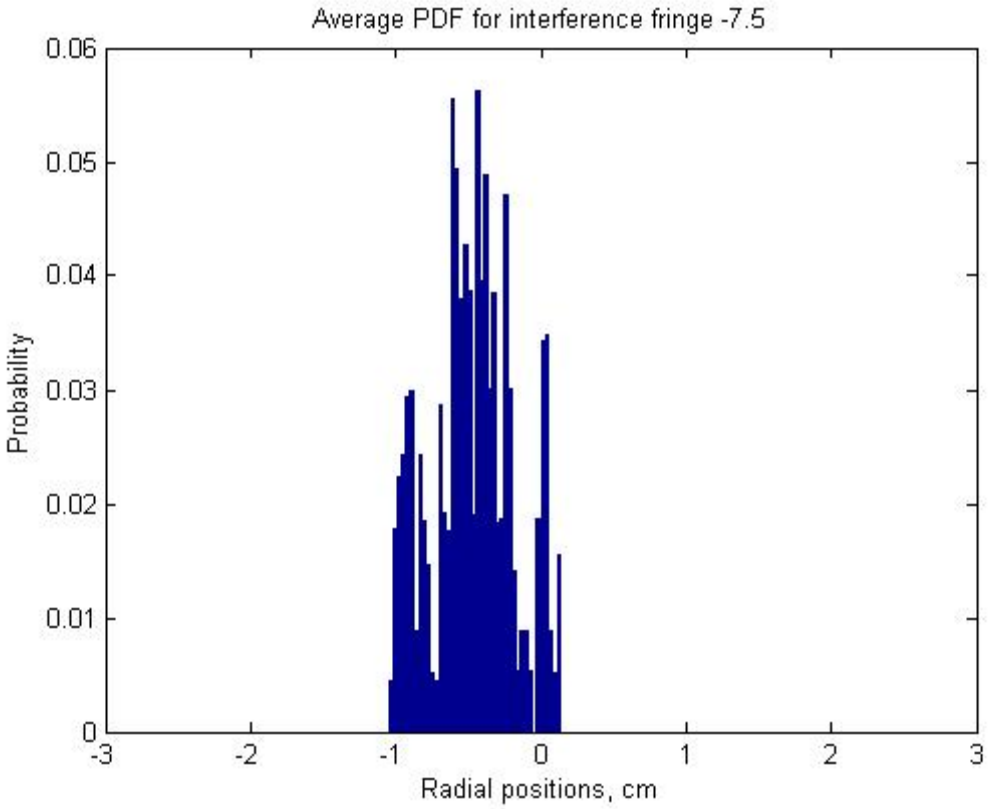


Figure 50: PDF for fringe -7.5 at x = 5 cm

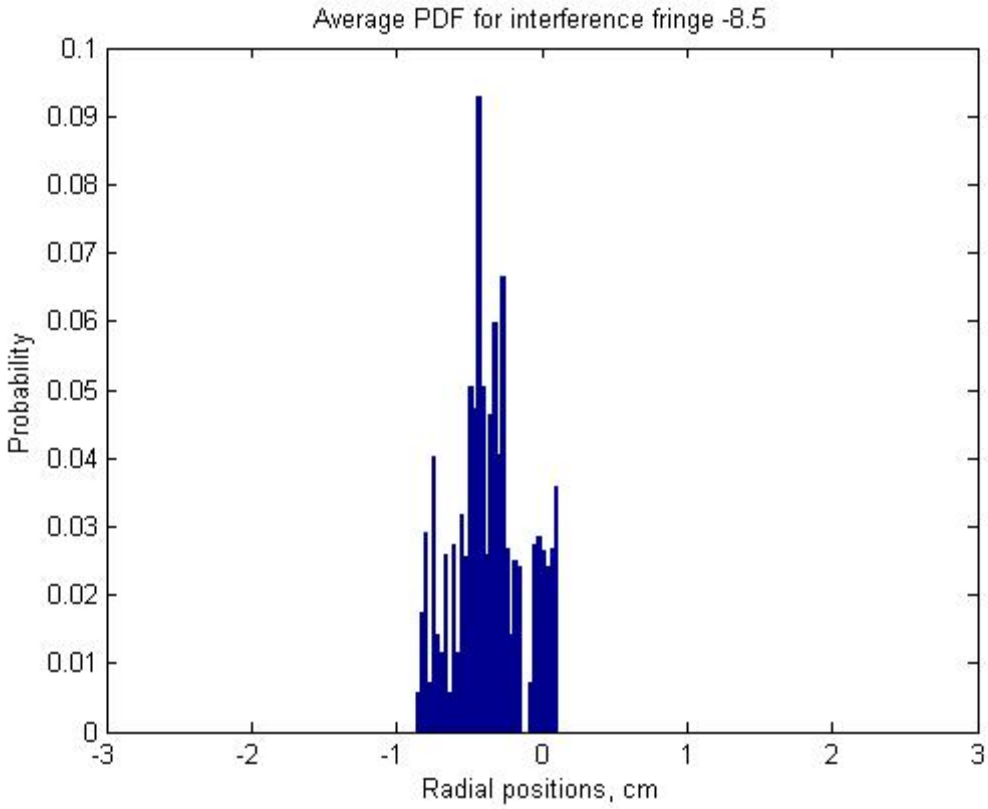


Figure 51: PDF for fringe -8.5 at x = 5 cm

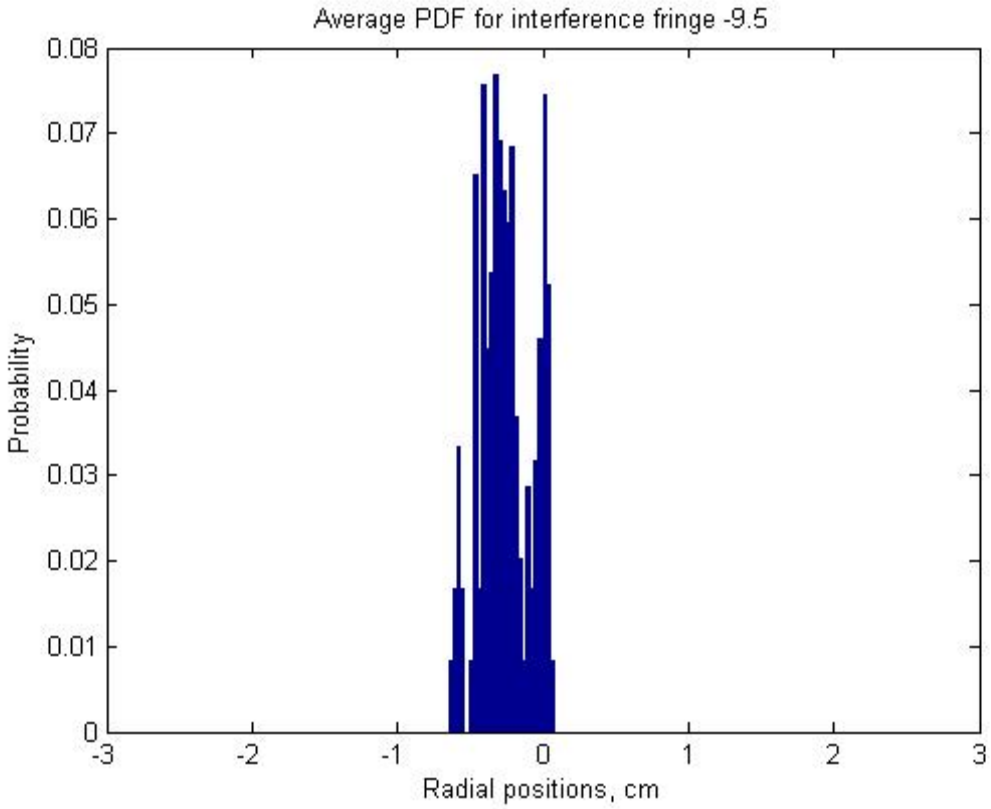


Figure 52: PDF for fringe -9.5 at x = 5 cm

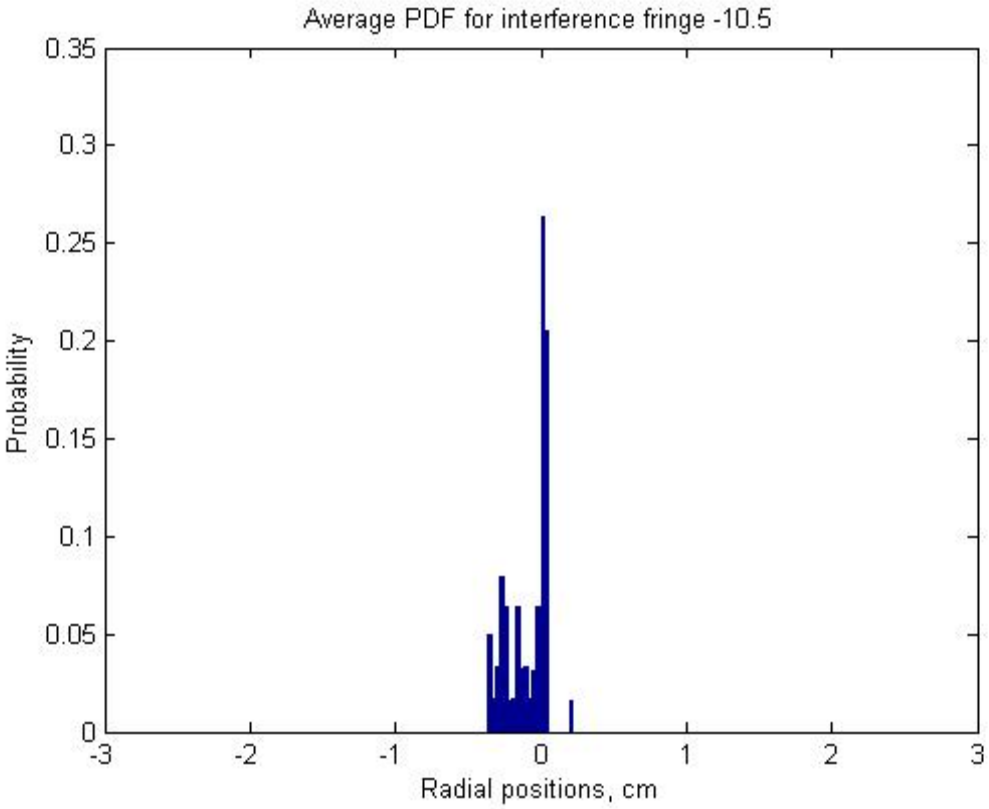


Figure 53: PDF for fringe -10.5 at x = 5 cm

REFERENCES

[1] S. Tieszen, "On the fuid mechanics of fires, *Annual Review of Fluid Mechanics*, vol. 33, pp. 67-92, 2001.

[2] B. Cetegen and K. Kasper, "Experiments on the oscillatory behavior of buoyant plumes of helium and helium-air mixtures," *Physics of Fluids,* vol. 8, pp. 2974-2984, 1996.

[3] D. Sharp, "An overview of rayleigh-taylor instability," *Physica D: Nonlinear Phenomena,* vol. 12, no. 1-3, pp. 3_10, July 1984.

[4] C. Herman, D. Mewes, and F. Mayinger (Eds.), "Optical techniques in transport phenomena," in *Advances in Transport Processes,* vol. 8. New York: Taylor & Francis Inc., 1999, pp. 1-58.

[5] C. Vest, *Holographic Interferometry*. New York City: John Wiley & Sons, 1979

[6] L. Hesslink, "Digital image processing in flow visualization," *Annual Review of Fluid Mechanics*, vol. 20, pp. 421-485, 1988

[7] A. Schoenbucher, B. Arnold, V. Banhardt, H. Bieller, H. Kasper, M. Kaufmann, R. Lucas, and N. Schiess "Simultaneous observation of organized density structures and the visible field in pool fires," *Symposium (International) on Combustion/ The Combustion Institute,* vol. 21, pp. 83-92, 1986.

[8] Z. Jiang, J. Liu, and G. Ni, "The calculation of the density field from axisymmetric schlieren interferograms by image processing techniques," *Acta Mechanica Sinica*, vol. 9, pp. 22-26, 1993.

[9] Z. Jiang and K. Takayama, "An investigation into the validation of numerical solutions of complex flow fields," *Journal of Computational Physics*, vol. 151, pp. 479-497, 1999.

[10] Z. Jiang, "Reliable validation based on optical flow visualization for CFD simulations," *Acta Mechanica Sinica*, vol. 19, pp. 193-203, 2003.

[11] R. Feeley, P. Seiler, P. Packard, and M. Frenklach, "Consistency of reaction dataset," *Journal of Physical Chemistry*, vol. 108, pp. 9573-9583, 2004.

[12] M. Haylock, N. Hofstra, A. K. Tank, E. Klok, and P. M. Jones, "A European daily high-resolution gridded dataset of surface temperature and precipitation," *J. Geophys. Res (Atmospheres)*, vol. 113, pp. 10-27, 2008.

[13] Wolfram|Alpha knowledgebase. (2010, August) July 10, 1984 Stuttgart, Germany. Online.

[14] M. Gawlowski, K. Kelly, L. Marcotte, and A. Schoenbucher, "Determining the effect of species composition on temperature fields using real-time holographic interferometry," *Applied Optics*,vol. 48, pp. 4625-4636, 2009.

[15] T. Kreis, *Handbook of Holographic Interferometry*. New York City: Wiley-VCH, 2005.